TYMNET as a multiplexed packet network

by JOHN KOPF

Tymshare, Inc.
Cupertino, California

ABSTRACT

TYMNET is a commercially successful computer network that has been in continuous operation since November, 1971. It contains over 200 nodes. Circuit routing is established by a central process, the supervisor.

TYMNET has never been fully documented in the literature. While similar to other packet networks, the mechanisms used by TYMNET differ significantly from those used in other documented networks, such as ARPANET. Packets are used, but only as the carriers for virtual channels between neighboring nodes, and are not themselves “switched.” The mechanisms used lead to good line utilization, especially for individual, small-volume transmissions. Flow-control mechanisms permit information sources and sinks to operate at different intrinsic speeds.

INTRODUCTION

TYMNET* is a communications network which was originally designed by Tymshare to handle its own communication needs, and is now operated by Tymnet, Inc. (a wholly owned subsidiary) as a common carrier service. TYMNET has been in service since 1971, during which time it has expanded from the initial 36-node network to the current 200 nodes. On the basis of both experience with TYMNET and new technological developments, TYMNET is currently undergoing a transition to a new, even more flexible network—TYMNET-II. TYMNET-I has never been adequately documented in the literature.

TYMNET has been described as a virtual circuit switching network, without setting forth the techniques used. In order to discuss these techniques, it is first necessary to give a general overview of TYMNET.

THE PARTS OF TYMNET

TYMNET consists of a large number of NODES interconnected by LINES. Only one line can connect two nodes. Each node has a description of the lines and neighbor nodes it has, as well as a local (partial) description of the circuits passing through or terminating at that node. The information in all nodes is the PRIMARY description of the network. TYMNET also has a SUPERVISOR, whose function is described below.

A node consists of a mini-computer, some simple interface circuitry, and the node-code. Nodes are classified by function. The function defines the interface, both in terms of circuitry, and also in terms of the code (in TYMNET, the interface circuitry is deliberately kept primitive, and the functional interface is built into the code).

The node-code can be functionally partitioned into several distinct modules:

a. Buffer management;
b. Supervisor communications;
c. Line interface;
d. One or more process interfaces.

One type of node is the TYMSAT, which contains the interface used for low-speed dial-up (asynchronous). It may also include an interface for high-speed access (bisync). A second type is the BASE, which contains the interfaces to TYMSHARE’s XDS-940, PDP-10, and IBM-370 computers. A third type is the TYMCOM, which is a base in function but interfaces to a broader variety of HOST computers.

The supervisor is a program which maintains control over the network and manages the network resources. The supervisor originally ran in a XDS-940, connected to the network through one of the bases, but has recently been re-coded to run in an INTERDATA 7/32, as one of the nodes in the network. There are several nodes in the network which can capable of running the supervisor, but only one supervisor controls the network at any given time.

When a supervisor becomes active, it has no knowledge of the topology of the network, or of any of the circuits active within the network. Its first activity is therefore to take over the network. This consists of taking over its node, determining the neighbors of that node, then taking over each of the neighbors, and iterating until there are no nodes left which have not been taken over. During this period, the supervisor also determines and records the

*TYMNET is a registered trademark of TYMSHARE Inc.
resources used by each circuit active in the network at takeover time. The supervisor retains this information as a secondary description of the network circuit topology, and will update this description on the basis of subsequent information volunteered by the nodes, and generated by the supervisor. If a part of the network is subsequently lost, such as by the loss of a line on the takeover path, the supervisor will retake all nodes which are still accessible through alternate paths in the network. The act of taking over a node generates a path from that node to the supervisor, so the node can communicate with the supervisor.

DATA WITHIN TYMNET

The basic unit of data is the CHARACTER or BYTE of 8 bits. A circuit within TYMNET is a construct which transfers a stream of bytes over a fixed path (and in both directions) while maintaining the order of the bytes. The content of the data bytes is transparent to the network, and is the responsibility of the interface at each end of the circuit. Because the sequence is maintained, the actual data need not really be 8-bit quantities, but may instead be anything from 1 bit to 1,000,000 or more bits—the only constraint is that the total transfer (including padding if necessary) be a multiple of 8 bits.

To contain the characters while in a node, BUFFERS are needed. In TYMNET, Buffers are assigned unique numbers, to distinguish them from each other. “BUFFER” in this context is actually a misnomer, since a buffer does not contain any data, but instead is a descriptor of where the data actually is, and how much data is there. Data bytes are actually stored in a separate area, which is allocated as the need arises, and which is freed when no longer in use. The “buffer” describes where the first byte is, how many bytes are there, and where the last byte is. In addition, a buffer descriptor also contains a pointer—unique to that circuit—which indicates which process is to take data from that buffer. By this mechanism, source and destination processes are uncoupled, and leads to complete flexibility as to circuit routing and termination. A byte is added to the buffer by appending to the last byte, and updating the buffer descriptor. A byte is removed by removing the first (original) byte, and updating the descriptor.

The actual implementation is as follows: Space is allocated for the buffer descriptors, buffer data area, and a linear bit array at the time the node-code is created. Primitive subroutines are provided to place and remove a byte from a specified buffer (descriptor). Each buffer descriptor includes a unique index into the linear bit array, and the routines are coded to manipulate the bit specified, turning the bit off if no data bytes are in the buffer, and on if there are one or more data bytes present.

The bit array is partitioned into areas, each of which corresponds to a given process. Each process can look at the set of bits assigned to it, to determine if there is anything to do, and if so, can pick one of the bits that is on, and translate the bit back into a buffer number. The bit order need not be in one-to-one correspondence with the buffer number order. When a buffer becomes empty, the bit goes off, and the process will ignore the buffer, until some other process places more data into it.

LINE PROTOCOL

TYMNET uses a line protocol which is essentially a variation on that used by other packet networks, but which has less overhead while providing flexibility. The normal TYMNET data packet (or physical record) has two bytes of header, from three to sixty bytes of contents, and four bytes of checksum at the end of the packet. With packets of variable length, line bandwidth is not wasted in transmitting padding or fill characters.

Each line is assigned a fixed number of channels. On a given line, a circuit is assigned to one channel, and has exclusive use of that channel for the duration of the circuit. At each end of the line, a pair of buffers have been assigned to that circuit, also fixed for the duration of that circuit. These channel and buffer assignments are made by the supervisor at the time the circuit is initially built, and each node along the circuit is informed of its portion. As a result, the line overhead is lowered, since complete routing information need not be transmitted along with each packet.

Each circuit is terminated at both ends by a connection to an interface. The interfaces include an ordered set of buffers, normally referred to as PORTS. The intermediate nodes along a circuit each have a set of buffers, assigned to passthrough usage. Passthroughs are in addition to, and distinct from, ports. The port interface has responsibility for the data protocol (e.g., code conversion, echoing of characters to terminals, hand-shaking across the interface, etc.).

The physical packet is subdivided into one or more subpackets (logical records). Each logical record contains two characters of header, which specify the channel number, and the number of characters of data present. This permits the multiplexing of a number of short messages (containing as little as one character each) into a single packet, with consequent savings in overhead, since the checksum at the end of the packet covers all of the logical records within the packet. Individual logical records do not need the overhead of individual checksums.

If single characters are traveling through a circuit, their overhead is high (2 bytes overhead/3 bytes of logical record). However, an interesting situation occurs dynamically. If, because of reduced bandwidth (due to a heavy load or line errors), bytes are delayed at some point in the circuit, the probability of a subsequent byte catching up increases. As soon as this happens, the bytes will start traveling as a pair (2 bytes overhead/4 bytes of logical record).

A round-robin algorithm is used to build logical records. That is, once a logical record is made for a given channel, that channel will not be serviced again until all other channels with data present have in turn been serviced.
When a logical record is made, the record will have available a certain number of bytes of data to fill out the packet. If there are less bytes than this present, they will all be placed in the packet. If there are more, as many as possible will be placed in the packet. This means that each circuit will have equal frequency of attention, independent of the actual data transfer rate. The user typing one character every 10 seconds gets as good service as the second user who is transferring 500 characters/second.

Experience has shown that the terminal user tends to get back 10 characters for each one he types. The overhead on the individual characters he types are high, but the characters appear infrequently. The characters he gets back, since generated by a computer at a high rate, tend to come back in large logical records, thereby enhancing the efficiency.

When the originator of a data stream is not limited by mechanical processes (as in terminals), and instead can introduce bytes into the network as fast as they can flow through the network (as is normally true of the HOST computers connected to the bases), then the tendency is to make a full packet containing the data for only that circuit. In this case, the overhead for the packet is [8 bytes (packet overhead)+2 bytes (logical record overhead)]/8 bytes (total overhead)+58 bytes of data], or 8/66 (=12.1 percent). By comparison, the WORST case is the transmission of a packet with only 1 byte of data (=90 percent), but this occurs only in the case where the line is so lightly loaded that there is only the one character to transmit, so we can afford the overhead. In a heavily loaded worst case, where we have a single byte to transmit for each possible channel, each channel will require three bytes of space within the packet, and we can thus multiplex 20 separate circuits in one packet. The overhead is now 46/66 (=69.7 percent), and if each logical record averages two bytes of data, we advance to 36/66 (=54.5 percent).

Actual measurements on lines carrying a normal mix of large and small data records indicate that the average overhead for a line is 1/6; that is, for each six bytes transmitted, five bytes are user data.

The "life" of a packet consists of being built, transmitted over exactly one line, and, when validated, being torn down (the contained data being "scattered" to the appropriate buffers). Because there are no variable delays, such as would be encountered in a multiple-link transfer, the packet can be acknowledged as soon as it is received, while the succeeding packet is in transit. This permits the use of a very small span of discrete packet identifiers. In TYMNET, the numbers range from 0 to 7, which may be specified in a 3-bit field. An acknowledgment is thus also a 3-bit field, and is piggybacked on the packet header of packets going the other way. The transmitter is constrained to never advance more than four identifiers beyond the last packet acknowledged. If the condition does occur, the oldest packet not yet acknowledged will be re-transmitted, until acknowledgment does occur. The acknowledgment specifies the most recent, sequential packet successfully received, and thus may acknowledge from 0 to 4 packets. Because there are never more than four packets in circulation, the amount of buffering required for packet storage in TYMNET is greatly reduced over conventional packet networks.

CIRCUIT CONSTRUCTION

Circuits are built at the request of a user. A circuit may only be built to a base, but may originate at either a TYMSAT or BASE port. The process will be described for a circuit originating at a TYMSAT, since this will permit the demonstration of some of the interface functions. A circuit origination from a base is conceptually the same, but varies in implementation.

Consider a user who dials up a TYMSAT node. The TYMSAT answers the call, but knows nothing of the characteristics of the caller. A message is transmitted to the terminal, requesting that the terminal be identified. The message assumes that the terminal is a 300 baud ASCII terminal. The user now types a single character which is used both to determine the terminal speed, and to identify both the parameters and processes to use for this terminal. (E.g., an ASCII terminal will have a different process than an IBM 2741, because the character codes are different, as well as the interaction mechanisms.) As soon as the terminal is identified, the message "please log in" goes out to the terminal, and the port is placed into a log-in mode. The user types the account name, an optional destination (host), and a password. This stream of characters is transmitted to the supervisor, along with information as to the node and port originating the information. The supervisor examines and validates this information, and, if any errors are found, the user is informed and placed back in log-in mode (errors may be bad account name, bad password, or destination inaccessible). If no errors are found, the supervisor examines the network topology for the best path from the originating node to the destination node, taking into account a variety of parameters. (Note that the destination specified is a host, not a node. This independence permits the free movement of hosts from one base to another if the situation demands. A host which has hardware failures can be readily moved to standby hardware, even if on a different base.) When the best path is found, resources are allocated all along that path, and appropriate messages are sent by the supervisor to each node along the path as to the linkages required for that node. The resources allocated by the supervisor consist of the channel on each line along the path, a pair of buffer descriptors for each intermediate node (passthroughs), and the port buffers at the destination. In addition to building the circuit, the supervisor also places information into the input buffer at the destination, concerning the account name, origination node and port, and terminal type, for the host. Once this is completed, the supervisor has no further interaction with the circuit—other than avoiding the re-use of the resources—until the session finally terminates, and the supervisor is informed that the circuit (and thus the resources used by the circuit) is no longer in use. When a user logs out, or is disconnected from the destination for any reason, the interface at the
TYMSAT reverts to the log-in mode, and he can then hang up, or log back in to the same or a different destination.

NETWORK INTERCOMMUNICATION

In any large, distributed system, a mechanism must be provided for intercommunication. In TYMNET, this intercommunication takes two distinct forms. Each line always has two channels preallocated for network usage.

One of these is used only for communication between a node and its neighbor. The information transferred in this case concerns such parameters as the additional traffic the node can accept on each channel on that line. The data path for these circuits is only one line long, and may only indirectly be passed further.

The second form is used for supervisor communications. As the supervisor takes over a node, the direction toward the supervisor is recorded, and the effect is that takeover builds a tree-shaped path to each node. All communication between a node and the supervisor is over this path, and as the network changes, this path is dynamically re-established.

The supervisor and nodes communicate by sending messages back and forth over the supervisor control path. Messages going to the supervisor are distinguished by one bit from those coming from the supervisor, and all messages include a field which specifies the destination (or origination) node. Messages from the supervisor consist of several different types, and perform functions such as building circuits, reading (and changing) locations in the node memory, placing characters into a buffer, and other housekeeping functions.

Messages originating at the nodes include streams of log-in information, responses to supervisor commands, and information concerning changes in the network (such as circuit termination, line errors, and host state changes, etc.). On the basis of this information, the supervisor modifies its description of the network, and the usage of the network resources. For example, if a line goes out, the supervisor may have lost control of part of the network, has probably lost some circuits (thus freeing resources distributed through the network), and has lost some choice as to alternate circuit routing. In addition, the loss of that line may have made some possible destinations inaccessible, thereby restricting the range of choice on logging in.

When the supervisor loses part of the takeover tree, it discards the secondary descriptors for the nodes lost, and then searches the neighbors of the remaining descriptors for nodes it does not have. These nodes are re-taken through the alternate, remaining nodes, and new descriptors are built. These reflect the circuits existing through those nodes at the time they are re-taken, but the existing circuits are not affected by the temporary loss of supervisor control. The only effect is the inability to build new circuits during this time.

The same process is used when a new node comes up on the network. When a node comes up, one or more lines connecting it with the rest of the network start carrying data, and the supervisor is informed. The supervisor will not use a line for two minutes after it comes into service, since the line may be bad, but after the two minutes are up, the line is declared usable, and if the node is not yet known to the supervisor, it is taken over. This capability permits the network to operate in the face of extreme failure rates, such as a large storm with massive line-outage and power failures.

FLOW CONTROL

A network circuit is a pipeline. If no buffering is involved, the input rate is equal to the output rate, and the pipeline is rigid. Once buffering is introduced, the input rate no longer need be the same as the output rate, and the pipeline becomes "soft", able to accept input at a greater rate than output. However, there is an upper limit to this process, equal to the buffering capacity.

TYMNET permits the instantaneous input rate to differ from the output rate, but controls the average rate by a BACK-PRESSURE mechanism. Periodically, each node examines all the buffers associated with an input process, and if a buffer exceeds some threshold, the process is informed to stop inputting until further notice. For a host interface process, this consists of a message to stop output for that port. For line processes, it takes the form of a bit-string sent over the line periodically, with a one-to-one correspondence between bits and channels, telling the neighboring node which of the channels can transmit more data. For a terminal interface process to a device such as a cassette player, this may take the form of transmission of XOFF and XON characters to the terminal.

Because of back-pressure, the host need not concern itself with details of terminal operation, such as the need for padding characters to account for carriage return delays, which are terminal dependent. The host simply starts filling the pipeline, and, if the output rate is lowered because of added padding, or because the terminal is very slow, eventually the back-pressure backs up to the host, and will henceforth throttle host output to the level of terminal output. The same mechanism is applied if a cassette terminal is inputting to a host which is so heavily loaded that it cannot accept data at the rate the cassette can generate it. This is even more important as the data-origination capacity goes up, as for a high-speed terminal or another host.

CONCLUSION

TYMNET has been in continuous operation since November, 1971, and through continuous evolutionary development has grown to a 200 node network. During this period, it has successfully handled all of the problems encountered, and shown an excellent level of reliability. Its commercial success and demonstrated abilities have led to its successor, TYMNET-II, being based upon the same general principles.
REFERENCES
