Routing and control in a centrally directed network

by JOSEPH RINDE
Tymshare Inc.
Cupertino, California

ABSTRACT

TYMNET I is a centrally directed network with over 200 nodes interconnected in a topology that allows alternate paths between nodes in the network. Routing within the network is done by a central supervisor program, with full knowledge of network topology and network load. Within network nodes routing is table driven.

The supervisor communicates with nodes through a command tree that is built at network takeover time. The supervisor has no a priori knowledge of the network topology when it starts network takeover, and the topology may change while the supervisor is in control. The control tree is dynamically modified by the supervisor to accommodate the new topology.

INTRODUCTION

In 1969 Tymshare Inc. started the development of TYMNET I* to supply the communications needs of its growing time-sharing market. The design objectives were to replace hardware multiplexing gear with a more reliable and more versatile communications facility. Beyond these functional requirements, since it was developed to meet the demands of a commercial environment, the network also had to be inexpensive to implement and operate.

A centrally directed network of Varian 620 mini-computers was developed. When the network became fully operational in 1971 it consisted of 30 nodes and a central supervisor (with three backup supervisors) that ran on an SDS 940 computer system. The network has since undergone a prolonged evolution and has grown to 200 nodes (see Figure 1).

ROUTING

TYMNET I has been described as a virtual circuit switched network which includes ARPANET type packets as a subset of its data grouping capabilities. A virtual circuit is bidirectional and ties up some memory in the nodes that comprise the circuit, but line bandwidth is utilized only when user data is flowing through the network.

Routing within the nodes is done implicitly through routing tables called permuter tables. Figure 2 shows a virtual circuit in TYMNET I with all its associated permuter table entries and buffer assignments. Each permuter table entry points to one buffer of a buffer pair. One buffer of the pair is for incoming characters, one for outgoing characters.

Each physical record (analogous to a packet) traveling between nodes is a collection of logical records, each of which is associated with a virtual circuit. This allows the physical record overhead to be distributed over the data of multiple users. As seen in Figure 3, each logical record has a header specifying its logical record number (used to index the permuter table) and a count of the data bytes contained in the logical record. To describe the routing of data in the logical records we define

\[
\text{link } \{P(i,j)\} \quad \text{a connection between two adjacent nodes}
\]

\[
P(i,j) \quad \text{the } j\text{th entry in the permuter table for link } i
\]

\[
O(P(i,j)) \quad \text{the contents of } P(i,j) \text{ (a buffer number)}
\]

\[
O(P(i,j)) \quad \text{the other buffer number of the buffer pair}
\]

A physical record arriving on link A containing a logical record X would cause the characters in logical record X to be placed in the buffer \[P(A,X)\]. Note that the physical record processor need have no knowledge of the final destination of the data in a logical record. Even knowledge about the link on which these characters will leave is not explicitly known. \(P(A,X)\) fully defines the path the data is to follow, because (1) the buffer \[P(A,X)\] is a port buffer, causing the characters in it to be processed by a port driver (this could be a host port or a terminal port), or (2) there exists a \(P(B,Y)\) for some \(B\) different from \(A\) (though \(X\) and \(Y\) may be equal) such that \(O(P(B,Y)) = \text{O}(P(A,X))\). When the physical record-making process runs for link B it scans \(P(B,i)\), \(i=0, \ldots, n\). Each \(P(B,i)\) points to a buffer; the

---

* TYMNET is a registered trademark of TYSHEARE Inc.
other buffer of the pair (i.e. \(O(P(B,i))\)) is checked for a non-empty condition. This condition will be satisfied by \(P(B,Y)\) causing logical record \(Y\) to be created with the data in the buffer \([P(A,X)]\).

It is possible for data for a given virtual circuit to arrive at a node in two separate physical records, but leave that node in one physical record. This can happen since both arriving physical records will have a logical record \(X\), and the data in both logical records will be placed in buffer \([P(A,X)]\). It is also possible that data arriving in a single physical record will leave in two physical records. This is why we speak of a flow of characters in TYMNET I, rather than packet switching.

A physical record is a collection of logical records plus a physical record header (Figure 4). A cyclic record numbering scheme is used to facilitate acknowledgment of physical records correctly received and to detect and retransmit records which arrived incorrectly. Sixteen bits of vertical checksum and sixteen bits of diagonal checksum are used to check that records crossing a link are correct.

**ESTABLISHING A CONNECTION**

A user connects his terminal to a host on the network by dialing a local node. After typing a character to identify his terminal characteristics, the user enters his user name, host number and password. This information is sent, by the local node, to the network supervisor. The supervisor verifies the user's name in the Master User Directory (MUD) and then checks for a correct password. (Passwords are stored in the MUD only in ciphered form. To check a password given at login, it is enciphered and compared to the cipher in the MUD.) If the user did not specify a host number, his standard host number is taken from the MUD.

The supervisor now knows the node which the user called (origination) and the node to which he wishes a connection (destination). The supervisor then computes the minimum cost path for the virtual circuit. Each link in the network has a cost associated with it. This cost is a function of the link bandwidth and the presence of overload conditions (Table I).

**TABLE I—Link Costs in TYMNET I**

<table>
<thead>
<tr>
<th>Line speed</th>
<th>cost normal</th>
<th>cost overloaded one way</th>
<th>cost overloaded both ways</th>
</tr>
</thead>
<tbody>
<tr>
<td>9600 bps</td>
<td>10</td>
<td>26</td>
<td>42</td>
</tr>
<tr>
<td>7200 bps</td>
<td>11</td>
<td>27</td>
<td>43</td>
</tr>
<tr>
<td>4800 bps</td>
<td>12</td>
<td>28</td>
<td>44</td>
</tr>
<tr>
<td>2400 bps</td>
<td>16</td>
<td>32</td>
<td>48</td>
</tr>
</tbody>
</table>
Thus the minimum cost path, as computed by the supervisor, is an optimization of network resources, rather than a dollar cost. Once a path has been plotted, the supervisor allocates buffer pairs and permuter table positions in each node on the path to create the virtual circuit. Messages are sent to all nodes along the path to make the appropriate permuter table entries. This implicitly causes buffer assignments in the nodes. Nodes send an acknowledgment to the supervisor after making a permuter table entry. Once all the acknowledgments are in, the supervisor sends the user name, the user's status (from the MUD), the originating node number and port number, and the terminal characteristics to the destination node, plus a message to tell the attached host that there is a new login.

The host may now read the user name to verify that this user name is valid on this host. No further checking is required! All the security checking has been done on a host, with access restricted to network personnel, where not even the passwords are vulnerable to theft. This allows a host with minimal login security to be connected to the network, with confidence that only authorized people may log in through the network. Additional login security may, of course, be imposed by the host computer system.

**NETWORK CONTROL**

The network supervisor is a program that runs under a special time sharing system on an Interdata 7/32. The

For Node 112

\[
\begin{align*}
[P(\emptyset, 2)] &= 200 \\
0([P(\emptyset, 2)]) &= 0(200) = 201 \\
[P(1, 5)] &= 201 \\
0([P(1, 5)]) &= 0(201) = 200
\end{align*}
\]

For Node 5

\[
\begin{align*}
[P(\emptyset, 5)] &= 8 \\
0([P(\emptyset, 5)]) &= 0(8) = 9
\end{align*}
\]

For Node 1000

\[
\begin{align*}
[P(0, 2)] &= 5 \\
0([P(0, 2)]) &= 0(5) = 4
\end{align*}
\]
supervisor, like any piece of software, and the 7/32, like any piece of hardware, are subject to failure. Although failures (hardware and software) are infrequent (on the order of one every three weeks) the absence of a supervisor to build virtual circuits cannot be tolerated for very long. To deal with this problem, four potential supervisors exist in the network, only one of which is active at any one time. The active supervisor keeps the other supervisors dormant by sending “sleeping pills” to them at regular intervals. If the active supervisor fails, the operators at the network control center can immediately awaken one of the dormant supervisors. Even without human intervention, the dormant supervisors will notice the absence of the active supervisor when they cease to receive “sleeping pills.” The various supervisors have staggered sleep times, at the end of which they will awaken if no sleeping pills have arrived. Thus, in case of a supervisor failure one of the dormant supervisors will awaken and take control of the network.

It is possible for multiple supervisors to be trying to take over the network simultaneously. This situation is resolved gracefully by the less dominant supervisor going to sleep when it discovers the presence of a more dominant supervisor.

**NETWORK TAKEOVER**

In order to control nodes and carry out the supervisor's function of building virtual circuits, the supervisor must know the capacity of all nodes, their link capacities, the network topology and the value of every permuter table entry in the network.

A supervisor starting network takeover has no a priori knowledge of the network topology. The supervisor first sends a takeover command to its own node, and learns of that node's capacity, the capacity of its links, every permuter table entry in that node and the neighbors of that node on each link. The latter is the basis on which the supervisor discovers the topology of the network. The supervisor now sends takeover commands to each neighbor of its own node. As each of these nodes comes under complete control of the supervisor, each of its neighbors is checked to discover previously unknown nodes, and these are in turn taken over.

In this way the supervisor learns the topology of the network, all its capacities and (from the permuter tables) which resources are in use. The network takeover duration is approximately three minutes. The time determining factors are the number of nodes, the bandwidth of links in the vicinity of the supervisor, the number of permuter table entries and the connectivity of the network vis-a-vis the supervisor's own node. From a control standpoint the supervisor views the network as a tree. The control tree comprises a subset of the links in the network. The balance and depth of this tree is the measure of connectivity of the network in relation to the supervisor's own node.

If a previously non-operational link becomes operational it may reveal one or more nodes that were previously inaccessible. The supervisor would then extend the control tree by taking over these newly discovered nodes. On the other hand, if a link that is part of the control tree goes out, the supervisor loses control of the nodes in that subtree. The control tree is then rebuilt to regain control of all the lost nodes that are still accessible through the network topology.

In a large network like TYMNET I, good connectivity is important to maintain fast response time and minimize bandwidth overhead. A deep supervisory control tree will cause undesirable delay in delivery of supervisory commands. Even independent of the centralized control proper-

---

**TYMNET I Logical Record**

- **A**: 8 bit Logical record number
- **B**: 8 bit byte count
- **C**: As many data bytes as specified in **B**

**Figure 3—TYMNET I logical record**

**TYMNET I Internodal Physical Record**

- **A**: 5 bit Sync pattern
- **B**: 5 bit size count
- **C**: 3 bit Record number
- **D**: 3 bit Acknowledgement
- **E**: Logical records
- **F**: 2 16 bit checksums

**Figure 4—TYMNET I internodal physical record**
ties of TYMNET I, a loosely connected network increases user response time by increasing the network transit delay. This added network delay time has been observed in the network as a whole and to the limit on the number of links each node may have. The physical topology of TYMNET I is based on a topology generating program which employs simulations using accounting data to determine virtual circuit lengths and telephone costs. The average TYMNET I virtual circuit is 3.1 links long.

The original TYMNET I nodes allowed only three links per node. Presently TYMNET I nodes allow 16 links per node. The cost of more links is increased CPU and memory requirements for the node. However, even 16 links per node will soon be inadequate for the growth of TYMNET. One solution to this problem is the use of node clusters. A node cluster is a group of two or more nodes in close proximity, interconnected by an inexpensive high speed distributed memory transfer device. We call this device a "memory shuffler," and it is capable of data transfers at memory bandwidth rates. We plan a fully interconnected eight node cluster at TYMNET's Cupertino center, with 72 links to the rest of the network. This approach creates a logically very large node (including two supervisor machines) that does not change the fundamental logical structure of the network. (A connection through the memory shuffler is viewed by the node and the supervisor as a high bandwidth link.) With clusters at all three major TYMNET centers in the United States, the network can be kept well connected (average virtual circuit length of less than two links plus a memory shuffler link, and network takeover time of one to two minutes) for the foreseeable future. A prototype of the memory shuffler is already deployed in the network.

Eventually, network growth will reach a point where further schemes to improve connectivity will yield small returns. At that time a partitioned topology (similar to the telephone area code) will become necessary. Little work has been done on this since no existing computer network has reached a size requiring a partitioned topology. Using TYMNET's centralized control approach, each area can have its own supervisor. There could be a master supervisor to act as the focal point of all interarea communication. Alternatively, each area supervisor could communicate with its neighboring area supervisors, and interarea routing could be done by an adaptive routing scheme (this is not unreasonable if the number of areas remains small).

REFLECTIONS

The underlying principles of TYMNET have passed the test of time. The orientation of the network to terminal users, and specifically to the support of the full duplex terminal, has given TYMNET a great deal of flexibility. Full duplex is more than simultaneous bidirectional transmission. It implies character by character interaction and full echo control. These features are an integral part of the TYMNET design (e.g., logical records of one or more characters). The addition of host to host communication required only a subset of the facilities available (e.g., echo control is not needed). Other high speed communications requirements have also been accommodated with low overhead.

Since TYMNET I interfaces to a wide variety of terminal types, a host connected to TYMNET I can similarly be accessed by a wide variety of terminals, even though the host may only be able to communicate directly with one type of terminal. In this sense TYMNET I is a more effective means (independent of cost) of connecting terminals to hosts than direct dial telephone service. TYMNET I's ability to translate between character sets, so useful in the case of accommodating a variety of terminals, can be turned off to permit shipment of pure binary data through the network.

The success of TYMNET's centralized approach can be measured, in part, by the routing overhead (worst case 1.25 percent of a 9600bps link) and the accessibility, by terminal users, of all of TYMSHARE's hosts (27 SDS940s, 10 PDP10s and 4 IBM370s) solely through the network. This relieves the hosts of password checking responsibilities, and reduces capital investment by having only one communications interface for terminals. Another feature of centralized control is a network clock, kept by the supervisor. The clock is a radio receiver for station WWVB, a time signal broadcast by the National Bureau of Standards. Each host is informed of the current time when it comes up, keeping all hosts synchronized (a distinct advantage for accounting). The central supervisor is not only a good source of information, but provides a natural collection point for network diagnostics and network accounting data.

The evolution of TYMNET I from a 30 node private network to a 200 node value added carrier is an example of successful adaptation of new technology to expand a general design beyond its original goals. However, the design of TYMNET I reflects the technology available in 1969. Today's lower CPU and memory costs remove the justification of some original design decisions.

As the network grew, the technology of minicomputers advanced, and prices came down, more powerful CPUs were brought into the network and new node types were developed with more capacity, in both CPU and memory. The additional capacity was used to provide more links per node and more ports per node. More terminal types became supported (e.g., 120cps terminals and batch terminals of the 2780 and 3780 type).

In 1975 the supervisor running on the SDS 940 reached its capacity and was replaced by a supervisor running on an Interdata 7/32, reducing network takeover time from 15 minutes to 2.5 minutes. Network takeover time is currently limited by bandwidth and connectivity of the network.

The original TYMNET I nodes had 8K of memory, a limiting factor on the capabilities of the nodes. This alone required the bulk of decision making to be vested in the network supervisor (e.g., allocation of buffer and permuter...
table entries in the nodes). A direct consequence of this decision was the supervisor's need to have a copy of every permuter table entry in the network. With network growth, this has proven expensive in both supervisor memory requirements and the length of network takeover. All the evolution and growth of the last five years has not changed that basic division of decision making between the supervisor and the nodes. This division is no longer in tune with the available technology, but changing it requires such a fundamental change in the implementation of both the supervisor and the nodes as to necessitate an ongoing redesign of both. The external view of the network need not change with this redesign, however. As a result the deployment of TYMNET II will occur in 1977 without disruption of service to users of the network.

ACKNOWLEDGMENTS

The author owes a great debt to LaRoy Tymes, the designer of TYMNET.

REFERENCES