INTERCONNECTION OF BROADBAND LOCAL AREA NETWORKS

by C. Sunshine, D. Kaufman,
G. Ennis, and K. Biba

Sytek, Inc.
Los Angeles Operations
Culver City, CA 90230

ABSTRACT

Interconnection of multiple broadband local area networks to form an integrated packet transport system presents several challenges. To take full advantage of broadband systems, assignment of nodes to channels must be dynamic, leading to the use of a flat address space. Combined with the desire to avoid reliance on a central server or complex routing in packet forwarders, this addressing scheme leads to adoption of a controlled flooding technique to "discover" the best path to a destination node. This discovery procedure sets up a path through internetwork forwarders for use by subsequent packets to the same destination. This paper describes the design and implementation of such a technique in Sytek's LocalNet(TM) systems along with several refinements which increase performance and keep the worst case load for route discovery below a few percent of network capacity.

INTRODUCTION

While interconnection strategies for long-haul networks have been fairly well studied, interconnection of local area networks is in an earlier stage of development. Broadband LANs present a special challenge due to their support for multiple channels on the same physical medium. Unlike baseband systems with their relatively short distance and single channel limitations, broadband systems can support distances of 25 or more kilometers and numerous channels. The individual channels are typically operated as logically separate networks, allowing high total traffic across all channels while limiting contention within each channel.

In addition to different logical networks on the same broadband cable system, networks on different cables may also be interconnected. Each logical network may have its own independent (and possibly different) means for exchanging packets between nodes within that logical network. That is, each network has its own physical and link layer procedures for transmission medium access, packet framing, and contention resolution. Interconnection of logical networks to form a network system is accomplished via packet store-and-forward gateways operating at the network level in the protocol hierarchy.

To take full advantage of broadband systems, assignment of nodes to channels must be dynamic, with network interfaces able to move freely among channels (by tuning of a frequency agile modem). It is also desirable that user nodes have constant addresses by which they can be identified regardless of their current channel assignment. These goals lead to use of a "flat" address space for all nodes in a broadband system, rather than the hierarchical addressing (with explicit network number) often used in long-haul systems. The cost of this approach is greater difficulty in finding nodes (routing). The efficient solution of this problem forms the central topic of this paper.

Sytek's LocalNet(TM) system is the prototypical example of a broadband system of this sort [1,2,3,14]. The basic transmission medium in LocalNet is a standard CATV cable system with two-way transmission capabilities. User nodes (terminals or computers) are attached to the network through Packet Communication Units (PCUs) that implement all protocol layers and provide various interfaces to user nodes (e.g., RS-232 with virtual terminal commands for interactive terminal connection). Packet forwarders called Bridges interconnect different channels; similar units called Links interconnect channels with point-to-point lines to other cables. Bridges and Links are primarily designed to interconnect multiple channels on the same cable system, but otherwise these packet forwarding devices correspond roughly to the "bridges" and "long-distance bridges" discussed in [6]. Figure 1 shows a typical LocalNet configuration.

The purpose of this paper is to describe our goals in interconnecting broadband channels within a LocalNet installation, and the technique developed to meet those goals, called "discovery." We believe that the discovery mechanism is a novel approach representing a previously unexplored and attractive point in the multidimensional space of network interconnection strategies. We first provide some background on previous interconnection strategies, and then explain how our goals determined the general outline of our approach. We then

© 1983 ACM 0-89791-113-X/83/0010/0025$00.75

present the details of the discovery mechanism for route establishment and subsequent data transfer. We conclude with some observations on alternatives for future work.

PREVIOUS WORK

Some of the earliest work on network interconnection was done in the Arpanet community. The approach developed in the Transmission Control Protocol [5] and later in the separate Internet Protocol [12] called for hierarchical addresses (network/local) and packet forwarders (called gateways) between networks. To reduce storage requirements (and maintain individual network autonomy), the gateways route incoming packets only on the basis of the network portion of the destination address. For robustness, the gateways continuously exchange information to perform a distributed route optimization function; each arriving packet is routed independently (without regard to virtual circuits). Although this approach has worked well for some time, new demands for mobile or multiply connected hosts and the interconnection of large numbers of independent local networks have revealed some shortcomings [13].

The experimental Ethernet developed by Xerox adopted a very similar strategy with hierarchical addresses and packet forwarders routing only on the network portion of the address [4]. The more recent Ethernet standard specifies an apparently flat address space (over the entire universe of nodes ever manufactured), but Xerox has included an explicit network number within the internet layer protocol of the Xerox Network System [16]. Hence this system, too, functions much as the Arpanet: source nodes must provide the network address of the destination (perhaps by consulting a name server) which the packet forwarders then use to route the packet.

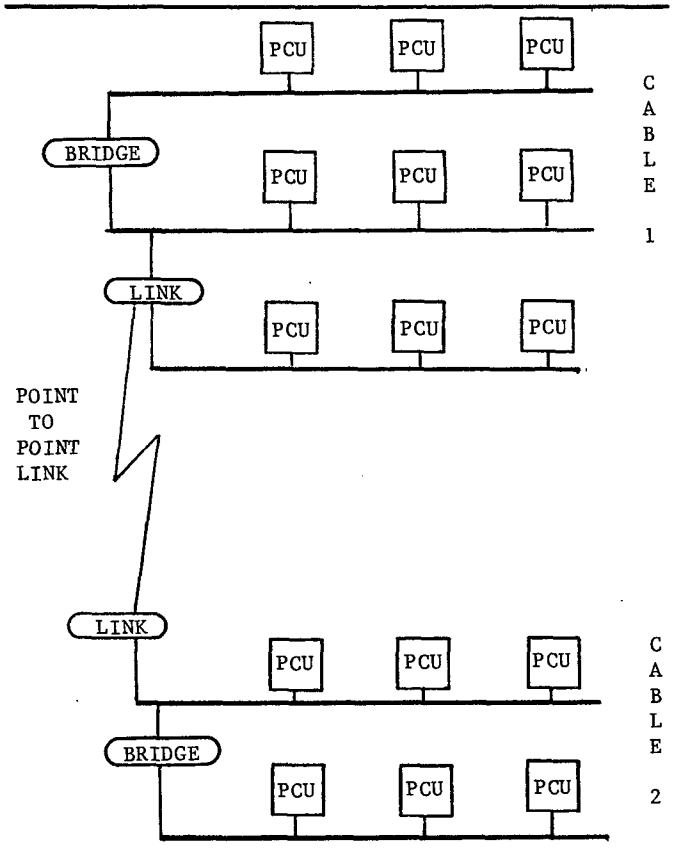

Figure 1. Generalized Architecture of a LocalNet Network.

Both of the above systems treat each packet independently and must make a best route determination for each incoming packet. Another possibility is to introduce the notion of a "path" into the packet forwarders. Paths may be computed in advance, set up by a central authority on demand, or set up by a special message sent from the source. In any case, the routing for that path is recorded in each packet forwarder on the path. Subsequent data packets then carry a path ID so that no new best route decisions need be made, just table lookups to determine the already recorded next link. This approach is used in several public data networks [15] and in IBM's SNA [9].

The benefits of path routing are decreased processing load in the packet forwarders on established paths, maintaining packet sequence (packets cannot get ahead by an alternate route), and decreased packet length (if the path ID is shorter than the destination address as in X.25 networks). The costs are increased table space, less robustness (due to the fixed route), and possibly greater processing load when paths are set up. Path routing is often tied to maintenance of virtual circuits between packet forwarders, but this need not be the case since error recovery, flow control, and sequencing on each link are separate functions from routing.

A final notion relevant to the LocalNet interconnection strategy is distribution of packets to all nodes by flooding or "hot potato" forwarding where an incoming packet is repeated over all links other than the one on which it arrived. Controlling this process to prevent an "avalanche" effect of infinite looping requires short term memory in each packet forwarder of packets recently seen so that duplicates received over a different path will not be resent. A procedure of this sort was developed to distribute routing information among the Arpanet IMPs [11]. While individual LANs often provide full interconnectivity via a broadcast medium, an interconnected system of such nets forms a graph structure much like the Arpanet which is not fully interconnected but does provide multiple paths. In the absence of a spanning tree or shortest path routing tables in the packet forwarders, both of which are difficult to maintain reliably [7], flooding provides a simple way to reach all networks.

GOALS FOR LOCALNET

As noted above, major LocalNet design goals have strongly shaped the development of our interconnection strategy. These goals reflect the unique combination of factors present in large broadband networks used primarily for session-oriented applications such as terminal concentration and terminal switching.

(1) Fixed addresses despite channel mobility. To take full advantage of broadband systems, nodes must be able to change channels easily (e.g., terminals tune to the channel to which the desired server is attached). It is highly desirable for nodes to have constant addresses within the system regardless of their channel tuning, which leads to the adoption of a flat address space with no explicit network or channel component. This allows users of the network layer to see a single integrated system without worrying about details of network topology or node location.

(2) No centralized server required. To maintain robustness and to keep costs low, it should be possible for any node to communicate with any other node in the system without requiring name or route lookup by a centralized server. Directory services are available as an additional option, but direct node-to-node calling capability is the basic service provided.

(3) Very large user population. Because of the relatively large total bandwidth available and long distances spanned on CATV systems, a single LocalNet system is able to support tens of thousands of users. Combined with requirement 1 for a flat address space, this means that unacceptably large databases (and update loads) would be required if the packet forwarders were to maintain knowledge of the location of all nodes. Thus neither a centralized directory (due to requirement 2) nor the Bridges and Links may be relied upon to determine best routes.

(4) Session oriented communication. The vast majority of user communication needs involve prolonged exchanges of information, so that establishment of sessions (or virtual circuits) should be the basic communication service provided. This means that some additional effort at session establishment time is worthwhile to minimize work required for data transfer during a session.

DETAILS OF DISCOVERY

This section presents the detailed operation of the discovery mechanism developed to meet the above goals. We first provide a brief summary of the mechanism, its place in the overall LocalNet protocol architecture, and some addressing issues. The full discovery procedure is then described, followed by some observations on key design elements.

Overview of Discovery

As noted above, the discovery mechanism is intended to establish routes between user nodes in a multi-network system without requiring knowledge of node locations or network topology by centralized servers or packet forwarders. Discovery is employed whenever a user requests a new LocalNet session. The creation of a session requires that a data connection be established between the user's LocalNet access device (PCU) and the requested destination user's PCU. The source and destination PCUs may be located anywhere within the LocalNet installation (i.e., neither their cable nor their current channel affiliation are known prior to the session request).

The first step of the discovery process is for the requesting user's PCU to broadcast a special Discovery packet on the channel to which it is currently "attached" (tuned). Upon receipt of the Discovery packet, each Bridge or Link attached to that channel records the discovery attempt and in turn broadcasts the packet on all other channels/links to which the Bridge/Link is attached. This process is repeated for all Bridges/Links in the LocalNet system and several tentative paths through the system may be built.

Eventually one Discovery packet reaches the destination user's PCU. While other copies of the Discovery packet may subsequently arrive via different paths, the destination PCU then sends a Response packet back along the path followed by the first Discovery packet received. The Response packet establishes that particular route as it returns to the requesting user's PCU. The route is then used for the duration of the session. All other tentative paths will be discarded after not being confirmed by a Response packet within a short time period.

LocalNet Protocol Architecture

Before presenting the detailed discovery procedures, it will be helpful to review the LocalNet protocol architecture [8]. A session layer protocol is used to manage connections between user ports on PCUs. The session layer depends upon the reliable, sequenced, and flow-controlled data transfer service provided by the Reliable Stream Protocol (RSP) at the transport layer. RSP is a connection-oriented protocol and therefore includes packet exchanges to open and to close connections. Each session requires exactly one RSP connection.

RSP utilizes end-to-end acknowledgements and retransmission in order to provide a reliable virtual circuit service. RSP assumes only that the lower layer protocols can (with high probability) deliver a packet to its ultimate destination based upon the destination address. The Packet Transport Protocol (PTP) is LocalNet's network layer protocol and it employs discovery to locate PCUs and subsequently utilizes the paths thereby established. Link protocols provide the basic capability to exchange packets across LocalNet channels or other communication links.

LocalNet Addressing

As mentioned earlier, LocalNet has a flat address space. Each PCU has a single 16-bit node identifier (or address) regardless of the PCU's location or channel affiliation. These node identifiers are the only addresses known to RSP and higher level protocols. Each PCU is also assigned an 8-bit link level address to take advantage of the address detection hardware of the PCU which can

only recognize 8 bits. For ideal efficiency, each PCU on a given channel would have a unique link address. However, upon receipt of each packet, PTP checks the node identifier to ensure that it corresponds to the PCU so that even if multiple PCUs on the same channel have the same link address, packets are only delivered if they have the correct node identifier.

Thus in order to send a packet efficiently, the PTP interpreter in the PCU must know what link address to put in the outgoing packet. Similarly, once a path has been established via the discovery mechanism, the Bridges and Links must know which link addresses to use in forwarding packets. A PCU when sending a packet along an established route sends the packet to the specific link address of the Bridge, the Bridge replaces this with the link address of the next Bridge, and so on. If PTP has not yet established a path, then the discovery mechanism is invoked and a link address of 255 is used meaning broadcast. All Bridges, Links, and PCUs will receive broadcast packets and deliver them to their PTP interpreter.

Discovery Procedures

Routing within a LocalNet system is essentially a two phase process. First, whenever a new session (and hence a new connection) is created, a route is established via the discovery procedure. Second, the routing data installed at the source node, the Bridge/Links, and the destination node during discovery is used to route packets while the connection remains open. The paragraphs which follow describe the procedures employed during each of the phases. The packet formats illustrated in Figure 2 indicate the fields used during discovery and subsequent route utilization.

Source Generation of Discovery Packet. The discovery mechanism is initiated any time a user requests a new session. When the session is requested, the RSP interpreter in the user's PCU sends a Connection Open message. The PTP interpreter piggybacks the discovery data on that message (see Figure 2) and broadcasts the resulting Discovery packet. The discovery data consists of the DiscoveryID, the TransactionID, and the Source Link Address together with the Source and Destination NodeIDs which are included in all packets. This data is used by Bridges and Links to detect duplicate Discovery packets and to establish the route as described below. The other trailer fields are set to null values. The TransactionID is unique to this (potential) session on this PCU. The DiscoveryID is set to one (zero is a reserved value).

The PCU saves the TransactionID and the DiscoveryID in a newly created "route table" entry, and awaits a response. If no response is forthcoming, the PCU times out and sends another Discovery packet. This procedure is repeated a fixed number of times; each time the packet transmitted is the same as the original except that the DiscoveryID is incremented. The PCU remembers the last DiscoveryID used. Processing of the response packet is described later.

Bridge Forwarding of Discovery Packet. Upon receipt of a Discovery packet, a Bridge/Link checks to see if the packet is a duplicate. Duplicate packets are discarded while new ones are forwarded on all Bridge/Link channels (except the channel from which the packet was received). A packet is identified by the Bridge/Link as a Discovery packet if the packet was broadcast to the Bridge/Link and if the Destination LogicalID (discussed later) in the packet contains the null value.

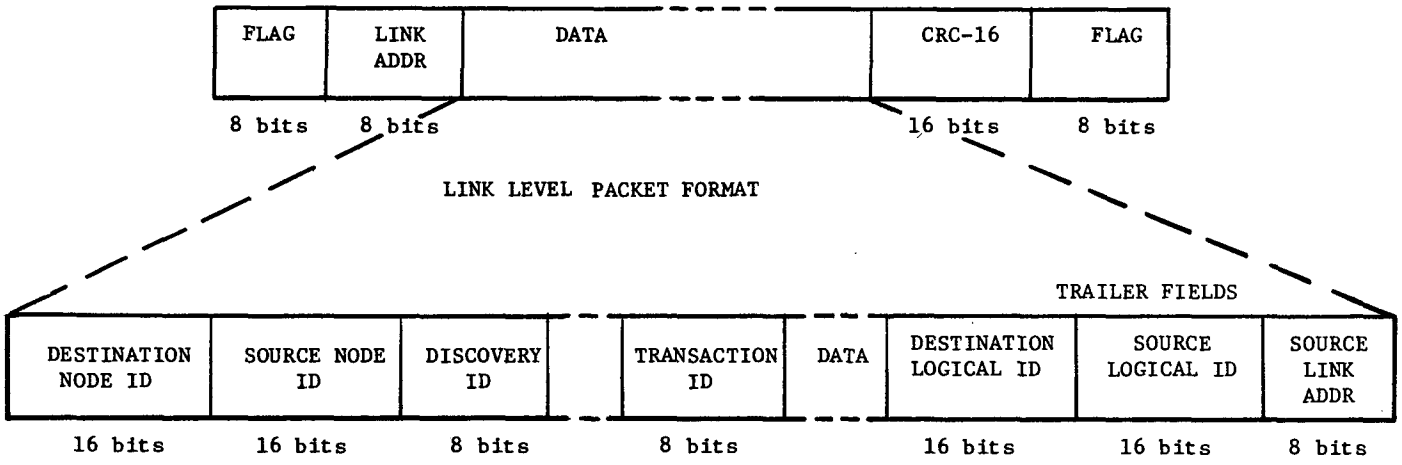

FIGURE 2. LocalNet Packet Formats

Duplicate detection is based on a data structure maintained by the Bridge/Link called the Discovery Pending List (see Figure 3). A Discovery Pending List entry is added each time a new Discovery packet is received by the Bridge/Link. Each entry contains the DiscoveryID, the TransactionID, and the Source NodeID from the packet as well as a timestamp indicating when the packet was received. The three ID fields are used to detect duplicate packets as follows: if the Transaction ID and the Source NodeID of a subsequently received Discovery packet are the same as those in an existing entry, and the DiscoveryID in the packet is less than or equal to that in the entry, the packet is a duplicate and is discarded. The timestamp is used to eliminate list entries after the maximum discovery lifetime has expired.

The Bridge/Link also maintains a second data structure called the Route Table which is used primarily for routing packets in the second phase after a path has been established. The Route Table contains a timestamp and a set of items defining the adjacent nodes (or hops) in each direction along the route (see Figure 3). The Source and Destination LogicalIds included in each Route Table entry are, in effect, pointers to the Route Table entries in the adjacent path nodes. A LogicalID is therefore a route or path identifier which is local to a given Bridge or Link. The timestamp is used to eliminate Route Table entries after "long" periods (i.e., about 15 seconds) of inactivity (because RSP sends a "keep alive" message at least that often on every healthy connection).

A new Route Table entry is created each time a new Discovery packet is received. The timestamp is initialized to indicate the arrival time of the packet; the location data for one of the hops is initialized with the Source NodeID, the Source Link Address, and the Source LogicalID from the Discovery packet as well as an indication of the source channel. That is, the entry now contains the NodeID of the initial source PCU and the location of the previous transmitter (possibly a Bridge/Link) of the packet. Thus the route is built "backwards" with routing information back toward the session initiator filled in from the arriving packet, and the route forward toward the destination left to be completed later.

Prior to forwarding the discovery packet, the Bridge/Link updates the Discovery packet trailer. The Source LogicalID is set to a pointer to the Route Table entry just created. The Source Link Address is set to the Link Address of the Bridge/Link interface used to broadcast the packet; a Bridge/Link may actually have different link addresses for each channel. The packet is broadcast on all channel interfaces.

Destination Processing of Discovery Packet. Upon receipt of a Discovery packet, the destination PCU determines whether the packet is valid. All packets are checked on receipt by all PCUs to insure that the Destination NodeID corresponds to the PCU's ID. Thus Discovery packets searching for other nodes are discarded immediately. Properly received packets are determined to be Discovery packets using the same procedure as the Bridge/Link (i.e., packet must have link address 255 and a trailer with null Destination Logical ID).

Duplicate detection by the receiving PCU also parallels the Bridge/Link: Any Discovery packet containing the same (or lower) DiscoveryID, the same TransactionID, and the same Source NodeID as another recently received packet is discarded.

DISCOVERY PENDING LIST USED TO DETECT DUPLICATE DISCOVERY PACKETS

DISCOVERY ID	TRANSACTION ID	SOURCE NODE ID	TIME
—	—	—	—
—	—	—	—

ROUTE TABLE CONTAINS DATA ON ADJACENT ROUTE HOPS

LOGICAL ID	TIME LAST USED	ADJACENT NODE 1				ADJACENT NODE 2			
		NODE ID	LOGICAL ID	LINK ADDR	CHANNEL	NODE ID	LOGICAL ID	LINK ADDR	CHANNEL
—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—

FIGURE 3. Bridge Data Structures

Note that the destination PCU accepts the first Discovery packet to arrive, thereby selecting the lowest delay route at the time the session is established, and rejecting later duplicate packets that arrive by longer paths.

The receiving PCU must be able to distinguish between attempts to establish a new session (and hence a new route), and repeated discovery attempts for the same session (if the Response packet is lost, the source PCU will try again). Packets with a new TransactionID represent the former, while packets with only a new DiscoveryID represent the latter. Each PCU maintains a set of route entries containing the Source Logical ID and the Source Link Address as well as the Discovery packet identifying fields. A route entry is maintained for each unique TransactionID and Source NodeID combination, also indicating the highest DiscoveryID received. The Source Logical ID and Source Link Address components of the route entry are always updated to reflect the most recently received valid Discovery packet.

The route entry at the destination PCU is used for all packets sent back to the requesting user's PCU. The first packet sent on this route serves as the Response packet and contains a trailer with information needed to complete the Route Table entries in intermediate Bridge/Links. The trailer includes the TransactionID for identification purposes and the Source Link Address. The Source Logical ID from the route entry is placed in the Destination LogicalID field of the trailer and the null value is placed in the Source LogicalID field.

Subsequent packets sent on this route contain trailers only if both PCUs are not on the same channel. While LogicalIDs are required by Bridge/Links to find the correct route table entry, PCUs on the same channel require only each other's Link Address and these values are exchanged in the Discovery and Response packets. The fact that the two PCUs are on the same channel is determined from the Source LogicalID in the Discovery packet. The originating PCU places the null value in this field, while Bridge/Links modify the field to a non-null value.

Bridge Processing of Response Packet. The Response packet appears to Bridge/Links as any other non-discovery packet. The packet contains the Link Address for the Bridge/Link's channel interface and the Destination LogicalID field in the trailer points to a valid entry in the Bridge/Link's Route Table. However, the Route Table entry does not yet contain any data on the source of the Response packet. Recall that the Route Table entry was initialized by the Discovery packet which contained data on its last hop. Naturally the next hop for the route was as yet unknown. Thus the Bridge/Link records in its table the channel on which the Response packet was received as well as the Source Link Address and LogicalID from the Response packet, thereby completing the Route Table entry for the path in both directions.

All non-discovery packets are then forwarded by the Bridge/Link based on the "other" hop data in the Route Table. The Bridge/Link sets the Destination Link Address in the packet header and the Destination LogicalID in the packet trailer in accordance with this data. Also, the packet's Source LogicalID is set to a pointer to the Route Table entry at that Bridge/Link; the Source Link Address is set to the Bridge/Link's Link Address on the channel over which the packet will be transmitted next. A minor optimization is employed on non-response packets. If the next hop is the last (receiver will be the PCU for which the packet is intended), the trailer is suppressed. This action parallels that of the sending PCU as described above.

Source Processing of Response Packet. Upon receipt of the response packet, the source (or originating) PCU completes its route entry. The Source LogicalID and the Source Link Address in the Response packet are recorded. Subsequent packets sent on this route will use the Link Address in the header and the LogicalID as the Destination LogicalID in the trailer. Again, if the LogicalID is null (PCUs on same channel) then trailers are not required.

Once a Response packet has been received, no further Discovery packets need be sent for this session. If the PCU has emitted multiple Discovery packets for this session (i.e., the DiscoveryIDs differ but the TransactionID remains unchanged), it does not affect the acceptance of the Response packet. For each Discovery packet sent by the requesting PCU, the responding PCU will send at most one Response packet (and will update its route at most once).

PERFORMANCE ISSUES

The discovery technique was formulated to satisfy a number of critical design criteria. Several of these, including use of a flat address space, support of node mobility on a per session basis, and independence from a central server have already been discussed. Performance is, of course, also a key concern. The paragraphs which follow briefly review discovery performance with regard to network overhead, switching efficiency, route selection, and network load balancing.

Since the discovery technique employs a flooding mechanism to locate nodes, care must be taken to avoid undue rebroadcasting of Discovery packets. The use of unique identifiers for Discovery packets enables Bridges and Links to detect duplicate discoveries. Thus each Bridge/Link will broadcast a given Discovery packet exactly once. The packet is broadcast once on the requesting PCU's channel and at most once per interfacing Bridge/Link on all other channels.

In order to ensure that duplicate packets are detected, Bridge/Links must retain their Discovery Pending List entries for the lifetime of the Discovery packet. Fortunately, the LocalNet packet lifetime is quite short, typically less than 0.5

seconds. PCUs can therefore use the simple algorithm of incrementing their DiscoveryIDs by one each time the packet is retransmitted and Bridge/Links can easily maintain a safety margin by retaining Discovery Pending List entries for several seconds.

The overhead imposed on the network by discovery traffic is quite modest. This fact can be verified by a simple calculation. For purposes of the calculation, we will assume a Discovery packet lifetime of one second and an average of two broadcasts of each Discovery packet on each channel (i.e., just over two Bridge/Link interfaces per channel). The overhead can be calculated for a single channel; the overhead as a percentage of total network bandwidth will be the same as the percentage overhead on the single channel.

We will assume that during the busiest 15 minutes (e.g., early in the morning, after the lunch hour, or following a power outage) a large number of new sessions are requested and that Poisson arrival can be assumed for the requests during that peak period. It should be noted that LocalNet PCUs employ a rotary call scheme to find an available port on a host supporting a large port pool. This scheme frequently results in multiple calls (i.e., to different PCUs in the same rotary) and hence in multiple discoveries; the average number of calls per session is 2.3 for rotaries at 75% utilization regardless of size. Finally, the calculation below assumes 2500 new sessions during the busy period and that channel utilization is kept below 70% (or 90kbps on the 128kbps channels).

avg session requests per sec = 2500/900 sec = 2.78
avg broadcasts per channel per discovery = 2.0
avg discoveries per session request = 2.3
=> avg Discovery packets per sec during the
peak period = 2.78 x 2.0 x 2.3 = 12.79

Since Discovery packets are 168 bits,
avg discovery load = (12.79 x 168)/90000 = 0.024

Assuming a Poisson distribution for arrival of Discovery packets,
avg Discovery packets per sec during the
peak period (with probability of .9999) = 29
peak discovery load = (29 x 168)/90000 = 0.054

Therefore, during the busiest anticipated period on a large LocalNet installation, the average load due to discovery is 2.4% and during the peak second is 5.4%.

The LogicalIDs included in packet trailers are integrated into the discovery mechanism in order to improve Bridge/Link switching efficiency. Once the route has been established and a steady-state achieved for the session, the Bridges and Links play the role of packet switches between channels. Each time a Bridge or Link forwards a packet, it must determine the appropriate channel and Link Address to use. The LogicalID which is selected by the Bridge/Link is used as a direct index into the Bridge/Link's Route Table. Since the efficiency of route lookup is a key factor in overall Bridge/Link performance we chose this ID swapping approach to maximize performance at a reasonable cost in Bridge/Link table size [9]. It should also be noted that since RSP guarantees that a packet is sent on every connection on a periodic basis, the Bridge/Link's overhead to manage its Route Table is minimal (unused entries may be timed out quickly).

By its very nature, the discovery technique provides best route selection and traffic load balancing. PCUs always respond to Discovery packets in the order received so the route selected is always the fastest route by which the Discovery packet traversed the network. Typically the route selected will be the fewest hop path between the source and destination PCUs. However, as the load on particular channels and Bridge/Links increases and as their nominal load thresholds are exceeded, the delay on those hops will increase. Therefore, assuming that multiple paths are available between PCUs, the total path delay for Discovery packets will vary with the load and alternate routes will be selected for new sessions.

CONCLUSIONS

The discovery mechanism implemented in Sytek's LocalNet devices provides a powerful and efficient mechanism for establishing routes between user nodes in a large, multiple channel, multiple cable, broadband local network environment where nodes are fully mobile over broadband channels and no central server is required. The discovery mechanism was optimized for a system where session oriented traffic is dominant and substantial effort is justified at session establishment time. The technique automatically provides best route selection and load balancing without requiring any connectivity or delay information to be maintained by packet forwarders, and accomplishes this with modest demands on network resources.

Sytek is also developing interconnection strategies for very large systems (more than 100 channels and 100,000 user nodes) where fully distributed discovery may not be feasible, or where directed tuning of calling nodes to a specific channel is desirable. In such systems a routing server combined with a source routing capability seems attractive, but many of the same path maintenance functions in Bridges and Links can be carried over so that both discovery and directed routing can coexist in the same system.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the participation of Mark Gang, Hemant Kanakia, Steve Butterfield, and Jon More in the development of the broadcast route discovery approach. Additionally, we acknowledge the contributions of Mark Gang and Dave Stevens to the refinement of the discovery technique.

REFERENCES

[1] K. Biba, "LocalNet: A Digital Communications Network for Broadband Coaxial Cable," Proc. COMPCON 81 Spring, IEEE, pp. 59-63.

[2] K. Biba, "Packet Communication networks for Broadband Coaxial Cable," Proc. Local Networks Conf., London, England, May 1981, Online Conferences, Ltd., pp. 611-625.

[3] K. Biba and J. Picazo, "Multiple Channel Data Communication System," U.S.A. Patent No. 4,365,331, December 21, 1982.

[4] D. Boggs et al, "PUP: An Internetwork Architecture," IEEE Trans. on Communications COM-28, 4, April 1980, pp. 612-624.

[5] V. Cerf and R. Kahn, "A Protocol for Packet Network Intercommunication," IEEE Trans. on Communications COM-22, 5, May 1974.

[6] D. Clark, K. Pogran, and D. Reed, "An Introduction to Local Area Networks," Proc. IEEE, November 1978, pp. 1497-1517.

[7] Y. Dalal and R. Metcalfe, "Reverse Path Forwarding of Broadcast Packets," Comm. of the ACM 21, 12, December 1978, pp. 1040-1048.

[8] G. Ennis and P. Filice, "Overview of a Broadband Local Area Network Protocol Architecture," accepted for publication in IEEE Journal on Selected Areas in Communication, November 1983.

[9] R. Jueneman and G. Kerr, "Explicit Path Routing in Communication Networks," Proc. Int. Conf. on Computer Communicaton, Toronto, Canada, August 1976, pp. 340-342.

[10] K. Maruyama and G. Markowsky, "On the Generation of Explicit Routing Tables," Proc. 5th Int. Conf. on Computer Communication, October 1980, Atlanta, Georgia, pp. 90-95.

[11] J. McQuillan, I. Richer, E. Rosen, "The New Routing Algorithm for the ARPANET," IEEE Trans. on Communications COM-28, 5, May 1980, pp. 711-719.

[12] J. Postel, C. Sunshine, and D. Cohen, "The ARPA Internet Protocol," Computer Networks 5, 4, July 1981, pp. 261-271.

[13] C. Sunshine, "Addressing Problems in Multi-Network Systems," Proc. INFOCOM 82, March 1982, Las Vegas, Nevada, IEEE.

[14] Sytek, Inc., "LocalNet 20 Reference Manual and Installation Guide," Doc. No. 2000-RM-382, March 1982.

[15] D. Weir, J. Holmblad, and A. Rothberg, "An X.75 Based Network Architecture," Proc. 5th Int. Conf. on Computer Communication, October 1980, Atlanta, Georgia, pp. 741-750.

[16] Xerox Corp., "Internet Transport Protocols," XSO28112, December 1981.