ISI Research Report
ISI/RR-87-180
March 1987

12

University
of Southern
California

Gregory G. Finn

Routing and Addressing Problems in Large Metropolitan-scale Internetworks

INFORMATION
SCIENCES
INSTITUTE
ISI
213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

AD-A180 184

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
This document is approved for public release; distribution is unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
ISI/RR-87-180

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
USC/Information Sciences Institute

6b. OFFICE SYMBOL *(If applicable)*

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS *(City, State, and ZIP Code)*
4676 Admiralty Way
Marina del Rey, CA 90292

7b. ADDRESS *(City, State, and ZIP Code)*

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
Advanced Research Projects Agency

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
MDA903 81 C 0335

8c. ADDRESS *(City, State, and ZIP Code)*
1400 Wilson Boulevard
Arlington, VA 22209

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
-------------	-------------	-------------	-------------

11. TITLE *(Include Security Classification)*
Routing and Addressing Problems in Large Metropolitan-scale Internetworks (Unclassified)

12. PERSONAL AUTHOR(S)
Finn, Gregory G.

13a. TYPE OF REPORT
Research Report

13b. TIME COVERED
FROM ________ TO ________

14. DATE OF REPORT *(Year, Month, Day)*
1987, March

15. PAGE COUNT
66

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
09	02	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
computer networks, network architecture, networks, network design, protocols

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Digital packet networking technology is spreading rapidly into the commercial sector. Currently, most networks are isolated local area networks. This isolation is counterproductive. Within the next twenty years it should be possible to connect these networks to one another via a vast internetwork. A metropolitan internetwork must be capable of connecting many thousand networks and a national one several million. It is difficult to extend current internetworking technology to this scale. Problems include routing and host mobility. This report addresses these problems by developing an algorithm that retains robustness and has desirable commercial characteristics.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. □ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL
Sheila Coyazo
Victor Brown

22b. TELEPHONE *(Include Area Code)*
213-822-1511

22c. OFFICE SYMBOL

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

ISI Research Report
ISI/RR-87-180
March 1987

University of Southern California

Gregory G. Finn

Routing and Addressing Problems in Large Metropolitan-scale Internetworks

INFORMATION SCIENCES INSTITUTE
213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

This research is supported by the Defense Advanced Research Projects Agency under Contract No. MDA903 81 C 0335. Views and conclusions contained in this report are the author's and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

Acknowledgments

The author would like to thank Al Beebe, Ron Castles, and Jon Postel for their help and suggestions. Also I thank Sheila Coyazo for the pains she incurred editing this.

Routing and Addressing Problems in Large Metropolitan Scale Internetworks

Introduction

Internetworking is an area of research that concerns the connection of packet-switching networks of different technologies. Much of the early work in this area began about ten years ago on the ARPANET [Cerf][Cerf-2]. The ARPANET conversion into the ARPA-Internet was completed in January 1983. An internet is constructed by connecting separate networks to one another via gateways [Sunshine]. Gateways are responsible for transparent encapsulation of networks and routing of packets that cross network boundaries.

The widespread geography of the ARPA-Internet was inherited from the original ARPANET. Its nearly 125 gateways interconnect up to 350 networks, the majority of which are local area networks. By comparison, a prototypical metropolitan area is a square 400 by 400 blocks. If one assumes that each neighborhood of four square blocks and each major office building may eventually have its own local area network, then a metropolitan-scale internetwork must allow the interconnection of 40,000 component networks and a national internetwork must interconnect several million. A network of this scale will carry very heavy traffic loads, which will place strong demands on any router to be fast and efficient. Against this background, we discuss some problems and limitations in existing internetwork design, and attempt to solve them by using a new form of address and a flat routing mechanism called Cartesian routing.

The Problem of a Network-Centered Address

Most internetworks assume that their routing task is finished when a packet is delivered to any gateway of the destination network. The internet does not route to a destination host, it routes to a destination network. The underlying assumption, that a network contains the best path to any of its hosts, has negative consequences.

Location Information and Poor Paths

An address is meant to describe a location. Internetwork addresses today identify a network, not a location. ARPA and Xerox Internet addresses contain implicit information about a location but no explicit information. From the address alone, one cannot determine the distance to a destination or in which direction the destination lies. When presented with a destination address in the ARPANET network, the Internet routes toward the nearest operating gateway into the ARPANET, not to the gateway nearest the destination host. If a network has significant geographic spread, the routing algorithm can produce poor paths to the destination.

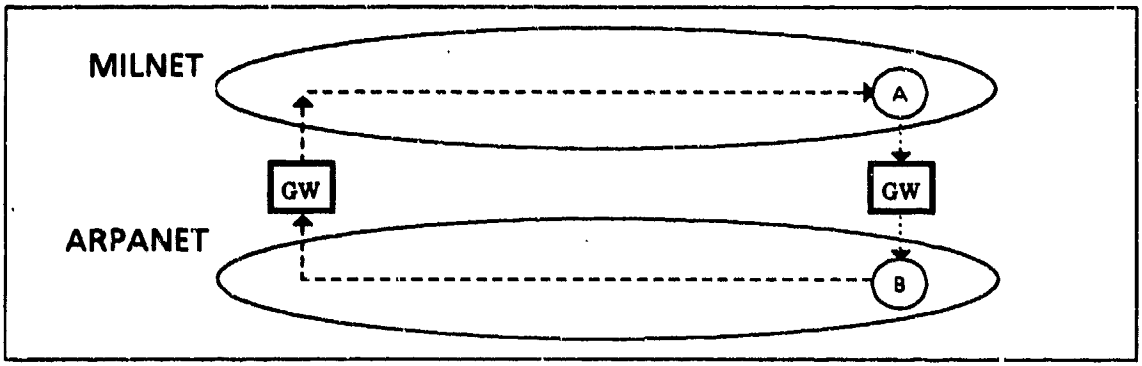


Fig. 1 Routing between networks

Figure 1 depicts what happens. Both the ARPANET and the MILNET are distributed across a continent. Assume that host A in the MILNET wishes to communicate with host B in the ARPANET. The ARPA-Internet routing algorithm considers path length to be the number of gateways between the source and destination networks. The true length of the path must measure the distance crossed within each network. Since this is not done, the routing algorithm considers both the dotted and dashed paths between A and B to have the same length. This results in the needless routing of packets between two geographically close hosts, back and forth across the continent.

Unnecessary Partitioning

If one assumes that a network contains the best path to any of its hosts, a problem occurs when a destination network becomes internally partitioned. Assume that host A wishes to communicate with host B and that both reside in YNET, which becomes partitioned as in Figure 2. Host B may appear unreachable from A, even though paths exist to it via other networks. Furthermore, any other host C will be unable to communicate with B if its path approaches B from the wrong 'side' of YNET. The internetwork has no knowledge of intranetwork topology and makes no attempt to avoid the partitioned region of YNET.

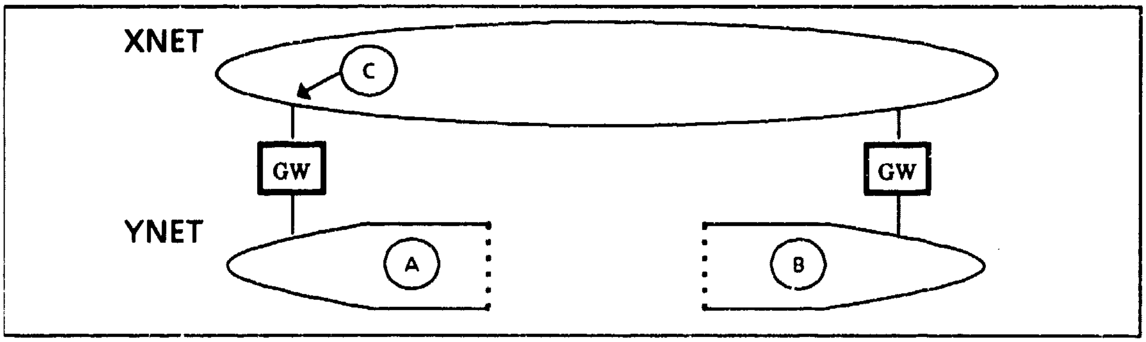


Fig. 2 A partitioned network

Host Mobility

The rapid expansion of cellular mobile telephones suggests that any pervasive public network must efficiently allow host mobility. Mobility is already important to certain business sectors; for example, delivery vans may contain display terminals. The network-centered form of address and routing algorithms in use today, which route toward networks, do not allow mobility across network boundaries. ARPA-Internet-style addresses strictly associate a host-ID with a particular network. The host-ID is not unique across networks. If a host is allowed to be completely mobile, this mobility unavoidably leads to confusion between the host's *identity* and its current *location*, since a mobile host generally crosses network boundaries as it moves. For the mobile host to function within the ARPA-Internet addressing style, it would have to become a host of each network it entered, changing its address as it moved. As a result, the DARPA research efforts in host mobility have been restricted to the creation of protocols for specialized networks that reside within the ARPA-Internet [Frank][Kahn]. This problem could be solved by a different form of address, in which location and host identity are logically separate.

Predictability of Service

Internetworks such as the ARPA-Internet are formed from a collection of cooperating, but often dissimilar, networks. Traffic from one network to another must in general pass through other intermediary member networks. There is no separate internetwork delivery system. However, using component networks as intermediaries on the path from source to destination causes service to be unpredictable. Topology changes cause packets to pass through differing sets of networks. A user does not generally know through which networks his packets will pass. Unless intranetwork loads, delays, and topology are measured as part of the routing decision, a user may see his connection data rate vary wildly over time.

Other severe problems exist if commercial operation is considered. Since packets may travel through different sets of networks during the life of a connection, accurate billing is prohibitively complex. Additionally, since traffic is routed through component networks, a customer may suddenly find his network used as a path for substantial amounts of unsolicited traffic. The customer may be denied necessary service on his own network. Both are intolerable situations.

This problem suggests that the internetwork should be based upon a two-tier system. The top (or transport) tier delivers packets between source and destination gateways. Its primary responsibility is data transport, and it acts as a common carrier. Once injected into the transport tier, a packet does not leave until it is either delivered to the destination gateway or discarded. Customer networks reside in the bottom tier. While customer-specific network-to-network connections may exist in the bottom tier, most long-haul communication would use the top tier. This tier is homogeneous, predictable, and isolatable, and it can be separately administered.

Protocol Design and Speed

Routers for a public network must be fast and inexpensive from a user's viewpoint. Today's routers are slow and expensive. The internetwork protocols in current use make speed difficult to achieve. Because the cost of communication is falling, faster non-local area communication linko are being employed. This is already causing problems.

T1 circuits (U.S.A.), at 1.544 Mb/s, provide 27 times the data rate of the ARPA-Internet's current 56 kb/s circuits. With an average packet length of less than 400 bits, one T1 circuit can produce 7680 packets/second, a rate of 130 µs/packet. The average number of memory cycles (instruction plus data) needed to process a packet in ARPA-Internet gateways approaches 3000 for typical microprocessors [Chiappa]. A 200 ns average reference rate is 20 percent of the required speed for just one fully utilized T1 circuit, and a router must realistically deal with several such simultaneously. This problem becomes very severe when considering emerging fiber-optic technology. In some areas, T3 fiber circuits are already available, running at 28 times the T1 rate.

Today's internetwork routers are based around minicomputer and microprocessor technologies. Even T1 rates seem outside the capacity of uniprocessors running today's protocols, and general-purpose parallel processors are much too expensive. To overcome this barrier, future internet protocols must allow routers to employ low-overhead, fast algorithms using specialized VLSI circuitry. A single chip or small chip set capable of processing packets from a half-duplex link is required. This solution eliminates algorithms which rely upon large routing tables or substantial amounts of computation per packet.

The Problem of Routing

Routing is a central issue in the design of any large internetwork. Routing has been approached as a graph theoretic question, and work in this area began at least as early as the middle 1950's. Key works are Dijkstra's discussion of the shortest path between two network nodes, calculated from the network distance matrix [Dijkstra], and the distributed routing algorithm of Ford and Fulkerson [Ford].

Centralized Routing

When the routing decision is centralized, one supervisory site or routing control center (RCC) determines the paths used from source to destination. This implies that the RCC knows with good accuracy the internet topology at all times and requires gateways to communicate changes in topology. In an internet with thousands of member networks, the topology can be expected to change frequently. This implies the following:

1. The algorithm that assigns routes must be run frequently, and the results distributed. For a large network, best-path algorithms require several seconds to execute.

2. Large amounts of traffic head to and from the RCC as gateways report topology changes and the RCC provides routing updates.

3. The RCC is a critical failure point. Backup computers must be available and strategies which determine which is the RCC must exist.

In certain networks, such as TYMNET, each connection's route is independently assigned by the RCC [Schwartz]. Predictably, the RCC becomes a bottleneck as the network grows. Centralized mechanisms do not scale uniformly and are wholly inadequate for large internets.

Distributed Routing

If routing decisions are distributed uniformly throughout the network, the obvious bottlenecks of centralized routing are eliminated. An early example of distributed routing was the original ARPA network routing algorithm [McQuillan]. The ARPA and Xerox Internetworks are the most prominent examples of this strategy today [RFC 823][Xerox]. They use modified *shortest-path* routing, producing a path between source and destination with the fewest gateway-to-gateway transitions. Each gateway accurately determines the next gateway on the shortest paths from itself to all member networks. The results are stored in a routing table. This is not true shortest-path routing, since intranetwork topology is not considered.

If the topology of the internet were unchanging, then gateway routing tables could remain static. However, in any realistic internet, gateways and communications links fail and networks may be added or removed. Since the topology of the internet changes with time, the gateways must have an adaptive mechanism for maintaining their routing tables. This is accomplished by exchanging routing table updates between active gateways. In the ARPA-Internet, adjacent topology changes are noticed within one minute [RFC 823]. In the Xerox Internet, this period is 90 seconds [Xerox]. When changes are noticed, an update is distributed to other concerned gateways. Distribution is event-driven in the ARPA-Internet and time-driven in the Xerox Internet, where updates are propagated every 30 seconds.

For the ARPA-Internet, update size depends primarily upon the topology of the network, as well as a number of other factors. However, for a large-scale internet, these updates would become very large. Figure 3 illustrates this for an x,y-grid topology. Assume that the link from node 1 to node 2 becomes inoperative. Both node 1 and node 2 will build a routing update and send it to their active neighbors. When an update is created, the gateway building the update creates an entry for any network to which it is as close or closer than the gateway receiving the update [RFC 823]. If gateway 2 is building an update to send to gateway 3, it must create entries for all networks at or below its own level (as indicated by the bold dashed lines). On the average, if there are N networks in the grid, the update will contain N/2 entries. Each entry is at least a few octets long, so if N=40,000, an average update will be over 60,000 octets long. The gateway update processing time is considerable and is $O(\mathrm{N}\log_2(\mathrm{N}))$, for large N.

A gateway sends update messages to its neighbors if it receives from some neighbor an update that is different than the one previously received from that neighbor. Thus an update from a gateway may initiate a cascade of updates. The shortest route from node 2 to node 1 is now longer, as is that from node 3 to node 1, and so on. By symmetry, we see that all the nodes directly above node 2 and those

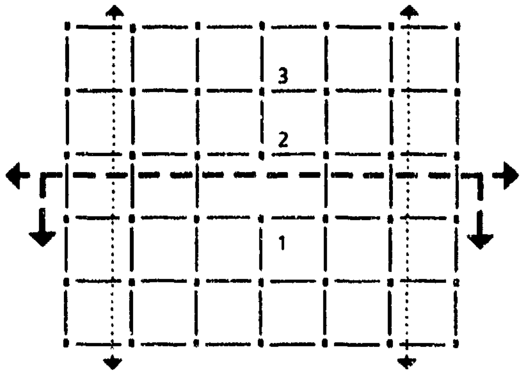


Fig. 3 Portion of x,y grid network

below node 1 must eventually receive new and different routing updates. Node 2 must send updates to those gateways on either side of it, as eventually must node 3, and so on. This results in an entire column of gateways receiving and generating routing updates. That column will be at least 5 gateways in width, as indicated by the dotted lines in Figure 3. As the internet grows, so does the frequency of updates, their size, and their distribution pattern. These sources of overhead sharply limit growth. This has led to some suggestions for decreasing the overhead.

Diminishing the Overhead of Routing Updates

Update length grows because gateways maintain a routing table entry for each active network. Since transmission time is only one portion of the overhead, increasing the link data rate is not a solution. Decreasing the frequency of updating slows response to topology changes and degrades performance. As an internet grows, traffic increases, and topology changes occur more frequently. Reducing the rate of updating as the internet grows is undesirable. An ideal mechanism would greatly reduce both the length of an update and the amount of time required to process it.

Routing by Destination Gateway

An alternative to routing by network address is to route by destination gateway address. This provides large improvements only if the ratio of gateways to networks is much less than one. Ratios of one-half or one-quarter would provide only a temporary improvement.

Two trends have emerged in network configuration. New organizations connect a single local network via a new gateway, tending toward one gateway per network. Older organizations connect new networks via an existing gateway, resulting in 2, 3, or more networks per gateway. These trends suggest that this method is impractical for update reduction, because the ratio of gateways to networks for the ARPA-Internet is only about one-third. This trend is not likely to change in a large internetwork.

Hierarchic Distributed Routing

The Xerox and ARPA internetworks treat all networks equivalently. The internetwork is a one-level addressing hierarchy above the member networks. This is a degenerate case of a multi-level addressing hierarchy [McQuillan-2]. Under this scheme, routing information improves as a packet approaches its destination. Progressively less routing information is available as the destination becomes more distant. This may result in great reductions of table storage and processing overhead for a router, since it need only maintain detailed data for nearby networks. Routers connected to local networks would be grouped into a cluster, clusters grouped into larger clusters or super-clusters, which in turn would be grouped into yet larger clusters.

The use of hierarchies to minimize overhead associated with routing table size has been analyzed by Kamoun and Kleinrock [Kleinrock]. Routing table length at the optimum is $\sim e \ln(N)$, where N is the number of networks. Each lowest level cluster ideally contains between two and three networks. Since the update table size grows as $\ln(N)$, the overhead due to transmission and processing of updates is kept low. Although shorter paths may exist between two arbitrary nodes, Kleinrock and Kamoun show that, as N grows large, the difference in path lengths between hierarchic and true shortest-path routing schemes approaches zero. However, like network-centered addressing, a cluster is assumed to contain the best path to any node in that cluster. If a cluster becomes internally partitioned, then nodes in that cluster become partitioned from the internetwork even when paths exist by routing outside the cluster. This problem is aggravated by the smallness of the optimal cluster size.

To minimize expected path length, the mechanism for choosing the cluster boundaries and cluster heads (nodes in a cluster that exchange routing information with neighboring clusters) should not be random. They should be chosen to simultaneously minimize intercluster distance and intracluster distance between a cluster head and the nodes within its cluster. The algorithm to do this optimally is NP complete and exponential in N. Burchfiel, et al., have presented a distributed heuristic approach to optimization, which is iterative and which converges [Burchfiel]. At each iteration, the algorithm requires each cluster head to receive information from all other cluster heads. Although guaranteed to eventually converge, convergence can be expected to take considerable time for a large internet with thousands of clusters. It is not clear how to apply this in a rapidly growing, dynamic network environment.

Limiting Update Propagation

Another tradeoff we might examine limits the extent to which routing update messages propagate. This limits the increased update traffic resulting from internet growth. Unfortunately, this approach introduces the risk of creating stable routing loops. Consider a series of gateways $G_0, G_1, G_2, \dots, G_n$, where G_1 is one hop from G_0, G_2 is two hops from G_0 and one hop from G_1, etc. Assume that this series of gateways forms the best path between two networks. Assume now that G_n loses contact with the destination network. Under the current ARPA-Internet routing algorithm, all $G_0, G_1, G_2, \ldots, G_n$ would be informed shortly of that loss and would recalculate their best path to that network. If routing updates propagated outward from G_n only to G_i, then G_{i-1} would continue to use the old path and route toward G_i. If G_i utilized directly or indirectly any of $G_{i-1}, \ldots, G_0$ as part of its new path to the destination, a stable routing loop would exist.

From this argument, it is clear that updates must be allowed to propagate from the source of the change to all concerned routers. It is also clear that temporary routing loops may be created as updates propagate; hence the speed of propagation is important. Temporary loops can be avoided by updating routing tables only after changed link information has reached all concerned routers [Merlin]. Knowledge that temporary loops are caused only by increasing link weights or link failures allows further refinements to be made [Jaffe]. However, avoiding temporary loops in this manner greatly adds to the complexity of the overall routing algorithms.

In an internet of unconstrained paths and topology, updated routing information must propagate throughout all gateways to guarantee avoidance of stable routing loops. In this sense, all updating mechanisms that avoid stable loops are equivalent. To limit piopagation of routing updates, some constraints must be placed upon allowable paths, internet topology, or both.

Cartesian Routing

We wish to develop an internetwork routing algorithm to solve the following problems:

1. Overhead that increases with size and thus limits expansion.

2. An inability to allow mobile hosts free movement throughout the internetwork.

3. Routing and packet processing requiring large tables or substantial computation per packet.

While these problems should be overcome, any new algorithm should preserve the desirable attributes of current algorithms: robustness in the presence of link and router failures, and the use of short paths.

In designing the routing algorithm, we have adopted the following philosophical position: Customers are best served by a network that provides the fastest and most economical transport. Where practical, other issues are subordinated to speed of transport. We desire fast, simple, and inexpensive routers.

In deciding whether or not to accept a new connection, virtual circuit routers negotiate with one another hop-by-hop. Virtual circuit routing has the great advantage of flow control, but its process of negotiation dramatically increases the software complexity of the routing procedure, and lengthens the period of connection initiation. Our requirement for simplicity rules out the use of virtual circuit routing. Therefore we shall adopt datagram routing.

We know from experience that the strategy of routing toward destination networks instead of destination locations inhibits the movement of mobile hosts. One of our principal goals in designing a new routing algorithm is to allow hosts to move freely, even across network boundaries. If we are to provide this kind of mobility, we must avoid the use of network-centered addresses. We require an address to specify a host's current location. (The address must also contain its identity, but the contents of the identity field are not of concern here.)

Shortest-path routing requires all gateways to have current knowledge of internetwork topology, implying the distribution of routing updates. This raises questions of scale. It is not yet clear how to adapt hierarchic shortest-path routing to a large and dynamically changing internetwork. However, it is clear that global distribution of updates greatly adds to router complexity; and for that reason, we wish to avoid shortest-path routing strategies. In abandoning shortest-path routing, we expect to trade some robustness for decreased overhead.

We associate a unique ordered Cartesian *location* with each gateway. We call routing based upon this type of address Cartesian routing. A Cartesian address is a two-tuple: <*location*, *id*>. The *location* element represents a position. Any coordinate system by which a metric distance can be calculated between two locations is sufficient. Latitude and longitude are most obvious. Hierarchic codes by which distance can be inferred via lookup are also permissible. We assume for the balance of this report that *location* is a two-tuple of <latitude, longitude>. A Cartesian address thus becomes <<*latitude, longitude*>, *id*>. The *id* is a local name, which varies from network to network. The *id* must be unique across the entire internetwork. Host mobility requires this; otherwise, two hosts with the same *id* could share the same *location*, and ambiguity would result.

Cartesian addresses make possible accurate billing. The use made by a customer of the transport tier is directly related to the amount of data transferred, multiplied by the distance that data is transported. The relevant information is immediately obtainable from each data packet. The addition of host mobility (discussed later) in no way complicates this.

We assume the presence of duplicate detection and retransmission in the connection protocol layers supported by the internet. A realistic model must also assume variance in topography, population, and service requirements (for example, irregularities caused by rivers and hills). First considered is routing in the absence of such irregularities.

Regularity of Interconnection

We assume the connection pattern between nodes is *Cartesian regular* in the following sense: Given any node (x_d, y_d) and any other node (x_s, y_s), an immediate neighbor of (x_s, y_s) exists, (x_i, y_i), which is closer to (x_d, y_d). There are many patterns regularly interconnected in the sense used here, most obviously an *x,y* mesh or grid pattern. We define *progress* of a packet from gateway (x_s, y_s) toward the destination gateway (x_d, y_d), as forwarding through any directly connected intermediate gateway (x_i, y_i) for which the distance $[(x_i, y_i)$ to $(x_d, y_d)]$ is less than the distance $[(x_s, y_s)$ to $(x_d, y_d)]$.

Pure Cartesian regularity is too restrictive. Within one hop of any node must be a neighbor closer to the destination. We define a hierarchy of successively less restrictive connection patterns. A network is said to be *n-hop Cartesian regular* if for any node (x_d, y_d) and any other node (x_s, y_s), some other node (x_i, y_i) exists within *n* hops of (x_s, y_s), which is closer to (x_d, y_d). *Progress* is now redefined as forwarding through any path from (x_s, y_s), of hop-count less than or equal to *n*, to a node (x_i, y_i), for which the distance $[(x_i, y_i)$ to $(x_d, y_d)]$ is less than the distance $[(x_s, y_s)$ to $(x_d, y_d)]$.

This definition suggests the following decentralized routing algorithm. If a network is *n*-hop Cartesian regular, a path between any two nodes can be found by proceeding initially as if it were 1-hop regular. When some node is reached for which no immediate neighbor is closer to the destination, a search of no farther than $(n-1)$ hops will find a node that has a neighbor that makes progress. Successive reapplication yields a path between source and destination. It is intuitively clear that any route composed of segments, each of which makes progress, cannot form a loop.

Response to Outages

Consider a router that encounters an outage on a path to the destination. Whatever the outage pattern, one can speak of its diameter measured in hops. Define the *outage diameter* to be the number of hops needed, from the point of encountering the outage, before progress is made toward the destination. In Figure 4, the dashed paths represent inoperative links. Figure 4a contains an outage pattern of diameter four. If the outage is encountered at router one, the shortest path (by hop count) that makes further progress toward the destination *d* (i.e., gets closer to *d* than node one) is [1→2→3→4→*d*]. In Figure 4b, from router one the outage diameter is five, via the path [1→2→3→4→5→6].

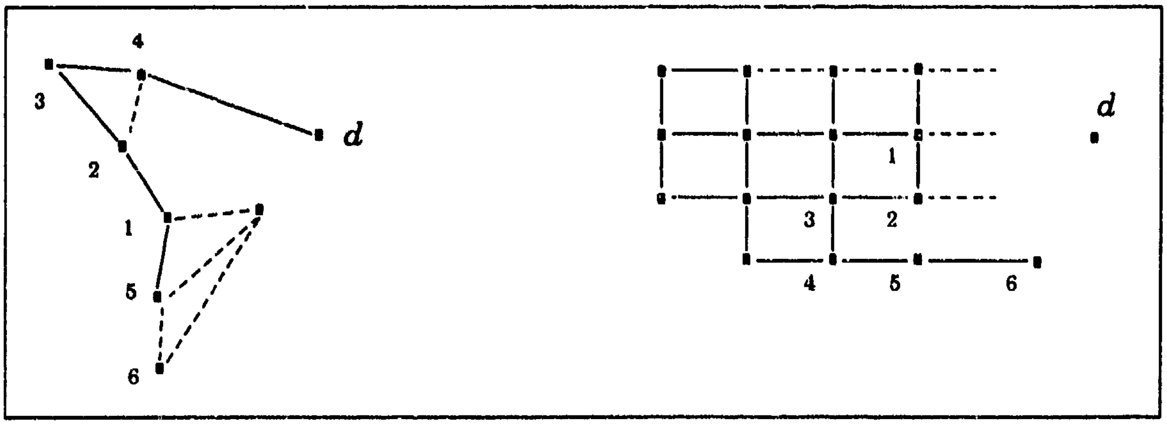


Fig. 4a Outage patterns Fig. 4b

A router that detects an outage or topological problem has no active link leading closer to the destination. Without knowledge of overall network topology, what approach can be used to route around most outage patterns? It is clear that an outage pattern of diameter two can be routed around if the remaining active neighbor routers receive copies of the packet. By definition, at least one of them will have a neighbor closer to the destination. This suggests that the router detecting the outage should initiate flooding, with distance restricted to one hop from the initiating router. Building upon this idea, an outage of diameter three can be routed around by the initiator limiting the flooding to two hops, and so on.

A priori, an outage's diameter is unknown. In general, the cost of flooding is exponential by hop-count. To be practical, the flooding limit should be small. However, there is a relationship between robustness, network topology, and the flooding limit. For a given topology and probability of outage, a flooding limit may be found that achieves a desired degree of robustness (an example is presented later). This limit is bounded from below by the Cartesian regularity of the network in a node's vicinity. Realistically, if the vicinity is *n*-hop regular, the flooding limit must be at least $(n-1)$.

A Sketch of a Limited Flooding Procedure

An initiating router adds to the packet header a *Flooding Flag*, indicating special action to be taken upon receipt by a nearby router, a *Flooding Limit Field* initialized to some limit n, and its own router address. It also places the distance between itself and the destination into a *Progress Limit Field.* The initiating router then sends copies to its neighbors. Upon receipt, a neighboring router determines whether or not it has active neighbors closer to the destination by examining the progress limit field. If it does, that router terminates the flooding procedure. A terminating router removes the flooding information from the packet and then forwards it. Otherwise, a router decrements the limit field. If the result is zero, the packet is dropped; if the result is positive, the modified packet is sent to all neighboring routers except the one from which the packet was received. The work of discovering an alternate route successfully concludes when a terminating router receives a flooding packet.

If the flooding packet keeps a trace of the routers transited until reaching the terminator, then a reverse path exists from the terminator to the initiator. The flooding packet header additionally includes an (n + 1) location *Flooding Router List.* Each non-terminating router appends its address to the list. The packet is forwarded to a neighboring router if the flooding limit field is positive and the neighboring router's *id* is not already in the list. A terminating router forwards the original packet. It also includes the address of the neighbor to which it forwarded the packet in the flooding router list and sends a *Flooding Reply Pccket* back to the initiator by means of source routing using the flooding router list that accompanied the packet. It also places in this packet the distance between the neighbor to which it forwarded the packet and the destination.

Upon receipt of a flooding reply, the initiating router creates an *Outage Table* entry. That entry associates an alternate source route with a destination address. Each time a router discovers an outage, the table is consulted. If an alternate source route exists, it is used. Otherwise, the limited flooding procedure is initiated. If an outage occurs in a source route, then the initiator must be informed by the source route router that detected the outage. This signals the initiator both that the outage table source route is bad and that it should start a limited flooding procedure for that destination.

Possible Problems

Responses to the flooding procedure may or may not arrive back at the initiator. The initiator must define a *Flooding Timeout* interval, which begins when limited flooding starts. If no replies have been received at the interval's expiration, it is assumed that no replies will be received. This timeout interval should be much less than any high level end-to-end connection timeout interval.

If the flooding procedure does not successfully terminate, the router assumes the destination to be unreachable. If it is successful, then at least one flooding reply packet is generated. That packet is either received by the initiator or it is not received. If it is received, an outage table entry is made. If it is not received, the limited flooding procedure is reinitiated upon receipt of another packet for the same destination. We assume that a flooding reply eventually reaches the initiator if the outage is of diameter less than or equal to n + 1. If the outage is greater than n + 1, we assume that higher level end-to-end functions inform a user of connection failure.

After limited flooding is initiated, a set of replies may arrive within the timeout interval. When the first reply is received, the source route it contains for the packet's destination is immediately entered into the initiating node's routing table. If more than one reply is received, which is best? This cannot be answered without complete topological knowledge, which is unavailable. Simulation suggests choosing the reply containing the source route that makes the closest approach to the destination. In the event that more than one reply makes a closest approach, choosing the reply with the fewest hops is reasonable.

We need not wait until the timeout interval has expired to make the choice. The best choice can be made incrementally, as flooding replies arrive. Candidate replies may arrive after the interval has expired. It is possible to choose one of these as long as at least one reply has arrived prior to the interval's expiration.

Routers may discover that an outage has been fixed. This may be due to the addition of a link, the return of a gateway to service, or the repair of a link. When this occurs, routers adjacent to the repaired outage consult their outage tables for any entries for which progress can now be made without the necessity of using a source route. Such entries may be removed from the table.

One element remains to be considered: an outage occurring on a source-routed path. A router receiving a source-routed packet has no choice picking the outgoing link. If that link goes down, the source route is inoperative. This situation is discovered when a router that is an intermediary in a source route receives a source-routed packet directing it to use an inoperable link. The intermediate router's response is to discard the packet and generate a *Source Route Outage* packet, using the original source route to construct a path back to the initiator. Upon receipt of such a packet, the initiator removes any entries that use the same inoperative path segment from its outage table. Subsequent packets for the same destinations will trigger limited flooding to find another path.

The flooding procedure as defined rapidly propagates a data packet. The data packet travels with the flooding request and is forwarded by all routers that successfully terminate the flooding procedure. As a result, multiple copies may arrive at the destination. An alternative procedure could be developed that holds the data packet until a flooding reply is received, and then forwards the data packet. This procedure avoids duplicate transmission but slows that packet's delivery.

Some Practical Observations

The sum of the routing and outage table lengths is bounded by N, the number of hosts. This is a severe drawback, but it can occur only in peculiar network topologies. In practice, the outage table length would be much smaller. However, practical mechanisms for achieving table lengths bound by a small integer are possible.

Outages due to topological irregularities are indistinguishable from those caused by a failed link. If a router wishes to send in a particular direction and cannot because there are no active links in that direction, it does not matter whether this is because links are down or because no links are laid in that direction. Topological irregularities are often large. A lake, river, or hill may not have network links crossing it. If these irregularities are larger than the practical limit to flooding, paths will not be found between source and destination. This suggests the periodic positioning of links of much longer point-to-point distance. These links would form a coarser mesh over the internet and could be expected to route around outage patterns of large diameter.

Large networks tend toward regularity of connection. Regular networks are simpler to maintain and to administer. However, in any large regular network the end-to-end hop count grows large (in a 200-by-200 x,y-grid the greatest shortest-path length is 398 hops). Each router-to-router transition causes a delay. That delay may be small, but for a large number of transitions the end-to-end delay becomes substantial. This is an issue regardless of which routing algorithm is chosen. It also suggests that some routers should have links of much greater point-to-point distance.

For example, overlaying the lowest level(0) grid with another level(1) grid covering $\sqrt{200} = 14$ average level(0) hops would exponentially reduce hop-count from 398 to ~56 hops (see Appendix). This solution produces a multi-level hierarchy organized by point-to-point link distance, with most routers linked only to a few immediate neighbors. A fraction of these possess additional links to routers quite some distance away. A fraction of those possess links to routers still farther away. The neighborhood routers comprise the lowest level, level(0), in a hierarchy. Above that are the level(1) gateways, above them the level(2) gateways, and so on.

A typical shortest-path algorithm will choose the path with the fewest hops. Cartesian routing by itself would not. Assume that each level(i) node knows shortest-path segments to neighboring level($i+1$) nodes. There exists a distance for any node in level(i) beyond which a path to the destination with fewer hops is typically found by employing level($i+1$) links for the majority of distance traversed. An overall exponential reduction in path length may be obtained by modifying the Cartesian routing decision to route up-level when the destination is farther than this limiting distance. This now implies that the routing tables in level(0) are bound by the number of hosts within the region circumscribed by the up-level limiting distance.

Routing Polygon

The *Outage Table* stores source routes which allow progress to be made toward destinations for which an outage exists. Until now, this table was treated as a vector indexed by destination location. That is sufficient to allow correct operation of the routing algorithm, but table size is bounded only by the number of distinct destinations within the radius set by the up-level limiting distance. In most circumstances a much tighter bound can be obtained.

An outage table entry contains a source route to another routing node. That node is a neighbor no farther than ($n+1$) hops distant, where n is the flooding limit. It is closer not only to the single destination which provoked the outage, but also to a set of destinations. In Figure 5b, two routers A and B are connected by a path of one or more hops. The virtual link between the routers is represented as a straight line from A to B. By bisecting that line, two half-planes are defined. A can make progress to any destination in B's half-plane by forwarding to B.

If A is in the interior of a network which is *n*-hop regular, then there exist routers no farther than *n*-hops distant which allow A to enclose itself in a polygon formed from the intersection of the lines defining such half-planes. Under this condition, A can guarantee progress toward any destination located outside the polygon. In general, there are many such polygons, and the polygon with minimal area guarantees progress to the greatest number of potential destinations. By implication, the smallest routing table which defines a polygon has three entries describing a triangle. In Figure 5a, A is not directly connected to C. To complete the routing polygon, a two-hop source route [A→B→C] is added. From the standpoint of limited flooding, any node in the routing table is considered a neighbor. Thus C is considered a neighbor of A.

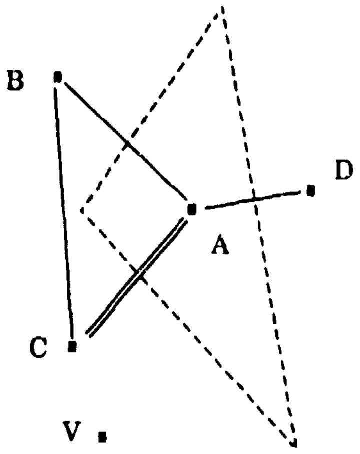


Fig. 5a

Routing Table for A

Node	Source Route
D	D
B	B
C	B, C

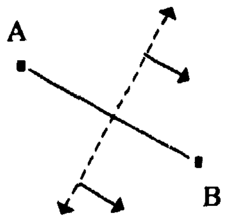


Fig. 5b

The polygon is built iteratively at router start-up, beginning with active immediate neighbors. Whether the polygon is closed is determined by summing the interior angles formed with A at the apex to the nodes in the routing table. In the example, initially only D∡B exists, which sums to ≦ 180°. This indicates that the polygon is not closed. To close it, A begins a limited flooding search for a hypothetical destination V, located within the region bounded by the exterior angle B∡D. (Since V is hypothetical, the flooding packet contains no data portion. Assume that a flooding terminator discards such packets after creating the flooding reply.) Assume as a result the source route [A→B→C] is returned. The interior angles D∡B, B∡C sum to >180°, closing the polygon. It is not always possible to close a routing polygon around a node. If a path to any node in the routing table becomes inoperative, then an attempt should be made to reclose the polygon.

If for a given location no closer entry is found in the routing table, then an outage is detected. This implies either that the routing polygon is not closed, or that the location lies within the polygon. This event triggers a limited flooding search. Assume the flooding limit is n. Any node in the routing table is considered a neighbor. Thus a search limited to (n − 1) hops proceeds from the nodes associated with each face of the routing polygon. Assume that this flooding meets with success. From the flooding replies, at least one reply is placed into the routing table. This adds a face to the polygon and decreases its area. If no successful reply is received, that destination is added to the outage table.

Packets for a destination in the outage table may be routed up-level in the hierarchy as a further attempt to avoid the outage. A higher level in the hierarchy presumably has a greater average point-to-point link distance and may be able to route around outages of large diameter. However, if the polygon is closed, the destination lies within the polygon; and for reasonable values of n and realistic local topologies, this step may be omitted.

The outage table now contains only entries for destinations which are closer than the up-level limiting distance and within a localized area of the network which is not *n*-hop regular. This is a tightly bounded region. Such a population is practically restricted, keeping the outage table small. The routing table need contain only as many entries as are necessary to close the routing polygon. This is the number of faces F. Similarly limited is the choice of source route to nodes for destinations farther than the up-level limiting distance.

Cartesian Routing: Pros and Cons

Typical characteristics of Cartesian routing may now be summarized. Because Cartesian routing scales indefinitely, it is appropriate for very large-scale application. It is very simple to implement and completely loop-free. It is also robust. For a given flooding limit n, all outage patterns $\leqq n+1$ in diameter are avoidable. Little state information is kept. The routing decision speed is O(F), where normally $F \leqq 5$. The response time to a change in topology due to a link failure is O(n). Storage costs are O(nF), since the source routes associated with each face of the routing polygon are bound by the flooding limit. The additional storage for routes to a higher level of a routing hierarchy (if they exist) does not alter the order. Cartesian routing allows nearly transparent inclusion of host mobility. It is also resistant to forms of attack that involve the supply and distribution of incorrect routing data. Some shortest-path algorithms are susceptible to this attack, which can shut down or partition their networks.

Cartesian routing does have drawbacks. It is not a shortest-path algorithm; shorter paths between the source and destination may exist, whether 'shorter' is defined to be fewer hops, less distance traversed, or lower delay. It is not as robust as shortest-path routing. If the outage diameter has size $>n+1$, then a path will not be found. Cartesian routing is not 'fail-safe' as is shortest-path routing, although reliability may be extended almost arbitrarily by increasing n. However, true shortest-path algorithms do not possess desirable scaling properties and cannot be used in a large internetwork. Modified hierarchic shortest-path algorithms do possess desired scaling attributes, but they assume that a cluster or network contains the best paths to any nodes within it. Thus a destination cluster can become partitioned when the vicinity is 2-hop regular. Robustness comparisons between Cartesian and modified shortest-path algorithms are therefore difficult.

Cartesian routing does not use shortest paths; although, with proper network design, it allows exponential reductions in path length to be achieved for 'long-distance' connections. Perhaps the largest drawback of Cartesian routing is its inability to respond to varying delays. However, a simple congestion control mechanism does exist. It uses a variant of limited flooding and adjusts to varying delay when that delay is due to congestion. Congestion control is described in a later section of the report.

Improving a Routing Polygon

Links added to a network due to expansion or repair may allow nearby routing nodes to close routing polygons or to decrease their area. The nodes at either end of a new or repaired link could use limited flooding to inform all nodes within the flooding limit of the link's existence. Consider routing nodes A and B. B reports a new or repaired link. As a result, A receives a flooding packet containing a source route from B to A. If the bisecting line between A and B cuts A's routing polygon, then inclusion of a source route to B in A's routing table produces a new polygon of reduced area. An alternative to this procedure would periodically remove any of A's routing table entries more than one hop distant, eventually forcing limited flooding by A to establish a new polygon. The former mechanism produces more rapid response while not interrupting service.

There are trade-offs between diminishing the area of a polygon and increasing path length. Assume that A has a closed routing polygon and receives a source route to B that allows it to decrease its polygonal area. Assume also that the decrease in area is small. If the route to B has a larger hop-count than those to the nodes associated with the faces of the polygon cut when including B, then adding B is normally not necessary. If it becomes necessary, limited flooding from A could rediscover it.

Maintaining Contact with Up-Level Routers

Routers are responsible for maintaining active source routes to nearby routers one level above. At initialization the addresses of nearby up-level routers should be known. Initial routes can be determined statically or dynamically. If a source route becomes unusable, a source route to another up-level router is utilized, while at the same time limited flooding is initiated to reestablish a source route to the previous up-level router. Each level (i) router should keep source routes to at least two level $(i+1)$ routers.

Determining the Flooding Diameter Limit

Assume that network topology is uniform and known with a good degree of accuracy. Assume also that routing node or individual link failures are independent events. We may then determine the limit for flooding, n, which achieves desired robustness.

Consider a network with an *x*,*y*-grid topology, as in Figure 6. If the probability of a routing node failing is <0.001, the probability of three colinear failures is

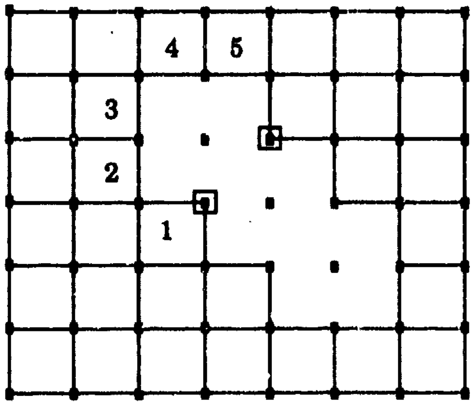


Fig. 6 Determining the flooding limit

$<1.0{\cdot}10^{-9}$. In this topology limited to three failures, a diagonal failure creates the largest diameter outage: five hops. Thus a flooding limit of $n=4$ is able to find a route around all outages of probability $\geq 1.0{\cdot}10^{-9}$.

Limits for Static Mixed Topologies

In the previous example, the network assumed a uniform *x,y*-grid topology with an average connectivity of four. In practice, however, the requirement of a uniform topology is unreasonable. In most cases, a network will have regions of differing connectivity, perhaps caused by topographic irregularities, where routing nodes are connected to fewer or more neighbors than the average. Under these circumstances, the calculation above must be carried out independently for each such region. The routing nodes in these regions would have flooding limits tailored to their particular topology, each ensuring the desired level of robustness.

Limits for Non-Static Topologies

Consider a network of unknown or rapidly changing topology. The methods discussed above are insufficient in this case. We appeal to the definition of regularity. Given any topology and the occurrence of an outage, the probability of an outage pattern of diameter n is greater than the probability of an outage of diameter (n + 1). This is true because at least one more routing node or link must be inoperative to increase the outage diameter by one hop (assuming routing node or link failures are independent events). Thus robustness can be increased for any topology by increasing the flooding limit. What that level should be requires careful judgment. There is a tradeoff of performance against the increased routing overhead necessary to assure the desired robustness.

Varying Limits for Differing Traffic Categories

Because the cost of limited flooding rises with the flooding limit n, it is not reasonable to set n arbitrarily high. The network designer should determine an acceptable level of robustness for normal traffic and set the default n accordingly. However, the use of a default does not mean that all traffic must restrict its flooding limit to n. Network traffic may occur in well-defined categories of increasing importance. If we assume that the network should try harder when delivering more important traffic, it is reasonable to increase the flooding limit when traffic is more important. An example would be unmarked traffic versus traffic marked urgent. Traffic marked urgent would have a higher flooding limit.

Flooding Limits and Broken Connections

The network designer may include a facility that allows either end of a connection to vary the default flooding limits along the connection path. A host at either end of a connection can receive a variety of replies indicating that the connection is no longer open, or the initiator may discover that it cannot be opened. One of these replies indicates that the destination is unreachable due to an outage on the path. The host receiving this reply may try to continue the connection or reopen it, with a larger limit for use with limited flooding. Under some circumstances this would allow a connection to be reestablished or opened when this would otherwise not be possible.

Position Error

To what extent does an error in claimed position cause incorrect forwarding of a packet? One serious error would be a stable routing loop, caused by routing nodes whose addresses reported incorrect locations. Assume that a routing node has an incorrect address location but aside from this follows Cartesian routing rules (see Figure 7). If routing node A has incorrect location information, it creates a virtual node A_V, which appears to its neighbors as if it is in a location it does not occupy.

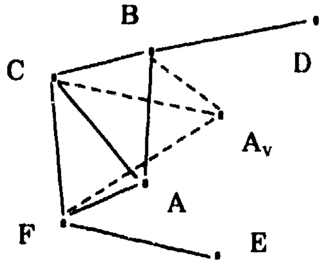


Fig. 7 Position error

If node C receives a packet for D, instead of sending it to B it will send the packet to A, whose virtual location A_V appears closer to D than B's location. This will not necessarily partition D from the network. Upon receipt of a packet for D, node A may or may not be able to forward the packet. Can a topology be constructed in which a stable routing loop is created?

A stable routing loop can only be created if a routing decision results in a packet's not making progress, if the packet is routed 'away' from the destination. If A claims its address is A_V, then node C sends a packet destined for D to A rather than to B, because distance $|D - A_V| < |D - B|$. Upon receiving the packet, A uses A_V in its distance calculations to determine to whom it should forward the packet. A cannot send it to B because $|D - A_V| < |D - B|$. Routing loops are avoided because A must still meet the progress criteria before forwarding a packet. By inductive argument, a chain of routing nodes, each reporting an incorrect location, will attempt to move a packet closer to its destination. At some point in the chain either the destination is reached, or a routing node which reports correct location is reached. In the latter case, either the packet can make progress, or a topological outage is detected.

Internetwork Reliability in the Presence of Incorrect Routing Information

Shortest-path routing algorithms achieve their behavior by allowing changes in topological information to quickly propagate throughout the entire internetwork. While this is a strength of shortest-path algorithms it may also expose the network to disastrous failure caused by the propagation of incorrect routing data or the excessive propagation of updates. There are several possible causes of this: (1) software error, (2) hardware error, or (3) deliberate supply of incorrect routing data.

Such a runaway event occurred in the ARPANET on October 27, 1980 [RFC 789]. As a result of a hardware error at just one site, the entire network became unusable for a period of several hours. It is important to realize that this occurred even though the other routers were operating correctly. The probability of such an event can be kept low through the careful cooperative design of router hardware and software, but it cannot be made zero. In a large commercial or military network, where the possibility of deliberate attack arises, the probability of such an event can rise dramatically.

In a Cartesian internetwork, routers do not propagate routing information throughout the entire internetwork. The Cartesian routing algorithm is also more resistant to deliberate attack than are some shortest-path algorithms. An error in claimed location can result in <u>localized</u> topological outages, where otherwise none would appear. The dispersal limit of incorrect routing information is controlled by the flooding limit. It is easy to place an upper limit on this so that any router encountering a packet with an excessive flooding limit or source route discards that packet. It would normally require the cooperation of many nodes simultaneously for a Cartesian network to become badly partitioned in this manner.

One may conclude, therefore, that Cartesian routing is relatively secure in the presence of incorrect routing data. This has important implications for any large-scale commercial application where customers connect to the internetwork with routers of diverse hardware and software implementations. The failure of an individual router's hardware or software cannot be allowed to halt the entire internetwork. Furthermore, the probability of such an event increases with both the internetwork size and the passage of time. Deliberate supply of incorrect routing data is more than a remote possibility in both large-scale commercial and military applications. The ability of Cartesian routing to resist such an attack is an important feature.

Heuristic Modifications

A number of heuristic modifications may be to the basic Cartesian routing procedures performed by a routing node. These heuristics improve some aspects of performance while retaining the limits on overhead already discussed.

Modifications to Limited Flooding

Building a routing polygon around a node creates a routing table that maps locations to source route segments that reach them. As in Figure 5, the source route segments may have length >1. The table for node A has the segment {B,C} for location C. We now modify the behavior of a node receiving a limited flooding packet.

Assume that node A has received a flooding packet and that A has no active neighbor which makes progress to the destination. If the flooding limit field is positive after being decremented, then under the original procedure A would forward copies of the flooding packet to its immediate neighbors after adding its address to the flooding router list. We can substantially strengthen the geographic coverage of limited flooding in some cases.

In Figure 5, A has an entry in its routing table for a remote neighbor C, accessed via the path [A→B→C]. C lies in a direction A cannot reach in one hop. We modify the flooding procedure and allow nodes such as A to forward flooding packets to all entries in their routing tables. In order to forward to C, node A would add the addresses A and B to the flooding router list and then utilize source routing to send the flooding packet to node C. Upon receipt, C would receive a flooding packet as if it were an immediate neighbor of A, but the routing list would take note of the extra hops used. A would not decrement the flooding limit field again, even though C is more than one hop distant.

The flooding router list would no longer be limited to $n+1$ locations, where n is the flooding limit; neither would the entries in a routing table, since they store results returned from a flooding procedure. Since this list could grow large in pathological cases, a limiting mechanism must exist. An implied limit is provided in the flooding packet by the number of locations available in the flooding router list. An additional restriction can be added as follows. If forwarding a flooding packet to a node (and thus adding an address to the router list) would exceed the flooding router list's maximum length, then the packet would not be forwarded to that node. This restriction allows an initiating node to control both the size of flooding replies and, by implication, its routing table entries.

A further refinement can be made, to extend the diameter of a search beyond the stated flooding limit under controlled circumstances. In Figure 8, x_1 attempts to reach x_d by initiating a limited flooding procedure. If it sets the flooding limit less than $(n-1)$, no flooding packet from x_1 will reach x_n. As a result, x_d will be unreachable from x_1. However, all nodes $x_4, x_5, \ldots, x_n$ do make progress toward x_d.

Ideally, flooding should be allowed to propagate beyond the flooding limit in such cases. Without complete topological knowledge, this is not generally possible, and the flooding limit is necessary to control the exponential cost of flooding. However, a compromise can be reached. In Figure 8, assume that the flooding limit $n=3$; then node x_4 will receive a flooding packet. If nodes that receive flooding packets decrement the flooding limit field only if the prospective next hop destination is *farther* from the destination than they are, x_4 would not forward the flooding packet to node y_1, thus significantly trimming the search tree. But x_4 would forward the packet to x_5; then by induction (assuming the routing list length limit is not reached), x_d will be reached.

With this heuristic, the flooding router list can again become large. It is restricted in the same manner as before. This heuristic predicts that paths tending toward the destination are more likely to reach it than those tending away. While this is not always true, in a densely connected network it should be true often enough to justify the heuristic.

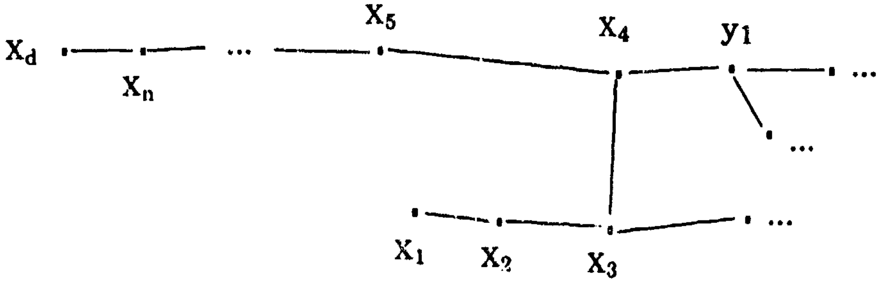


Fig. 8 Topology illustrating flooding heuristics

Partial Elimination of Route Retracing

Consider the situation in Figure 8. Assume that the network is n-hop Cartesian regular. Upon receiving a packet for x_d, node x_1 will request limited flooding to construct a source route $\{x_1, x_2, \dots, x_t\}$, $t \leq n$. Node x_1 will use this procedure when routing packets toward x_d in the future. Assume that a flooding reply is received. The flooding reply packet is seen by x_2 on its way back to x_1. Under normal circumstances, the routing table for x_2 implies x_1 as the initial hop for a packet destined for x_d. If x_2 examines the flooding reply, it can deduce that the subset of destinations for which x_t is closer than x_1 will be rerouted by x_1 back to x_2 as directed by the source route in the flooding reply. In this situation, x_2 may safely assume that the source route $\{x_2, x_3, \dots, x_t\}$ is a better path to x_d and enter that route in its routing table. Note that it does not remove any routing entry.

Once the entry is placed into x_2's table, subsequent packets sent to x_d follow a path two hops shorter than they otherwise would. The partial path $x_2 \rightarrow x_1 \rightarrow x_2$, which retraces itself, is eliminated. We extend this idea as follows. Consider a series of nodes $\{x_2, x_3, \dots, x_t\}$, and a node x_i, $2 \leq i < t$, for whom the routing decision for destination x_t is x_{i-1}. Upon receiving a flooding reply packet with source route

$\{\ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_t\}$, x_i may safely assume the source route $\{x_i, x_{i+1}, \ldots, x_t\}$ is a better path and place that route in its routing table.

Node x_{i-1} is a cul-de-sac from the point of view of x_i and the subset of destinations for which x_t makes progress. If x_i chooses the entry in its routing table which makes the most progress toward x_d, then this procedure removes a routing cul-de-sac of length $t<n$, where n is the flooding limit. After the flooding reply reaches x_1, the path lengths for these destinations are shortened by $\{2, 4, \ldots, 2t\}$ hops for packets initiated at nodes $\{x_2, x_3, \ldots, x_t\}$, respectively.

A network locality can learn to avoid a certain class of topological irregularity by using this technique. For networks that are throughout regular and richly connected, the elimination of route retracing provides little performance gain. In an irregularly connected network such as the ARPANET, however, simulation shows that this technique improves average performance by 10 percent.. This technique can be generalized.

Implicit Routing Table Addition

In response to a successful limited flooding operation initiated by node x_i for destination x_d, flooding reply packets are generated. These replies follow reverse routes, reaching the node that initiated the operation. An intermediate node x_k, on the route $\{x_i, \ldots, x_k, \ldots, x_t\}$, receives by implication a current source route to the set of destinations for which x_t is closer than itself. This is the subsequence $\{x_k, \ldots, x_t\}$ of the route beginning with itself.

At this point, the intermediate node x_k may safely enter that subsequence $\{x_k, \ldots, x_t\}$ as a source route in its routing table. Two possible cases arise:

1. No entry exists which makes progress toward x_d in its routing table. Either x_k's routing polygon is not closed, or the destination lies inside its polygon. In the former case, the source route adds a face to x_k's routing polygon. In the latter case, the polygon can be shrunk. If x_k subsequently routes to an element of the set of destinations served by the new entry, then it need not initiate limited flooding. This speeds initial communication with elements of that set.

2. There is already an entry that makes progress for the destination x_d in the flooding reply. In general, x_k will receive a subset of the flooding replies. These may include other source routes $\{x_j, \ldots, x_k, \ldots, x_m\}$. The choice it makes for inclusion into its routing table can be based upon whatever heuristic it decides is advantageous. Choosing the reply that is shortest by hop-count is the most historically satisfying, but it is not necessarily optimal.

Implicit Routing Table Deletion

Assume that an intermediate node x_k receives a packet containing the source route $\{x_i, \ldots, x_k, \ldots, x_t\}$. It has no choice of outgoing link. If the next-hop destination x_{k+1} in the source route is inoperative, the entire source route will be inoperative.

Any other source routes using the link $[x_k \rightarrow x_{k+1}]$ will also be inoperative. Ideally, x_k should initiate limited flooding in an attempt to deliver the packet, remove any entries in its routing table that use $[x_k \rightarrow x_{k+1}]$, and inform other nodes using that link to do likewise. The actions taken by x_k are as follows:

1. Strip the source route out of the packet and reroute it. This will most often require initiating a limited flooding procedure.

2. Remove from its routing table any entries that use segment $[x_k \rightarrow x_{k+1}]$.

3. Inform other nodes upstream from x_k on the source route that the source route is no longer usable. Node x_k creates a *Source Route Outage* packet containing the original source, the destination, and the inoperative source route segment $[x_k \rightarrow x_{k+1}]$. This packet is reverse source routed. Each node $x_i, x_{i+1}, \ldots, x_k$ along this path should take action (2) above.

Not all other nodes using link $[x_k \rightarrow x_{k+1}]$ will be informed by these actions. Only nodes along the source route of the packet that caused the outage to be discovered will be informed. The other nodes will be informed in a similar manner when they attempt to use that link. Node x_i will normally receive a Source Route Outage packet from x_{i+1}. Since it is at the head of the source route, it must immediately initiate limited flooding to obtain a new source route if possible.

Source Route Substitution

As in step (1) above, when node x_k notices that a link $[x_k \rightarrow x_{k+1}]$ is inoperative and a packet is source routed over that link, a limited flooding operation normally occurs. There are circumstances when this is not necessary. If x_k has in its routing table an entry that does not use the inoperative link and that makes progress toward the destination of the packet relative to the first node in the packet's source route, x_k may freely substitute its table entry for the source route. This action avoids the limited flooding operation.

Area Defined Broadcast

If addresses contain geographic position information, a new class of network broadcast communication becomes possible based upon area. The ability to circumscribe a desired region provides a feature much like a ZIP-code. This capability likely has commercial and tactical military benefits.

Assume that the source desires to reach all accessible hosts within a circumscribed area. It sends an *Area Broadcast* specifying a center (x,y), radius Δd, host address [*,*], and message content. Any routing node receiving the packet within Δd distance of (x,y) initiates a variant of flooding called *Area Flooding*. Copies of the packet are sent to any neighbor lying within the circumscribed region. This is a convenient way in which hosts providing a particular service could inform gateways in that area, and thus their attached host populations. It is also a mechanism whereby hosts within an area could be alerted to some event. If the host is known, the address [*,*id*] would attempt to send the packet to that host in the specified area. A related feature is an area-controlled query, which is described later.

Congestion Control

Congestion is one of the most serious causes of delays in the routing of information. Network-wide propagation of delay information is a commonly suggested solution to the problem of congestion, although even that solution has severe problems when applied to a heterogenous internetwork [IEN 189]. However, Cartesian routing does not propagate routing or delay information beyond a router's immediate vicinity. Because Cartesian routing avoids widespread propagation, that solution is not available to us.

Consider a congested router x_1. The external network cannot reroute congestion that occurs if x_1 is a source or destination for too much traffic. This occurs when an attached population of hosts overloads x_1. This type of congestion requires x_1 to deny service to a subset of its hosts, and it is not discussed here.

Congestion may also occur when x_1 is an intermediary, and when routes that use x_1 begin to carry more traffic. This type of traffic is expendable from x_1's point of view, since presumably another intermediary can be found. However, any congestion control mechanism that reroutes such traffic must take care not to reroute any traffic for which x_1 is a terminus.

Flooding Denial

Assume that x_1 is congested, and another router initiates limited flooding for destination *d*. It is unwise for x_1 to accept any new traffic for which it is not a terminus. If x_1 now processes limited flooding packets, those packets represent possible additional future traffic, since the set of flooding replies received by the initiator may contain source routes that use x_1. If the initiator chooses one of these routes, then x_1 will normally receive additional traffic. This can be prevented if x_1 drops all limited flooding packets it receives for destinations that it does not terminate.

This solution causes the flooding initiator to receive only a subset of the possible flooding replies. No reply is received for which x_1 is an intermediary. Unless x_1 is a terminus for *d*, only a source route that avoids x_1 could be chosen by the initiator. If the situation changes and x_1 becomes less congested, then x_1 may choose to again participate in limited flooding.

If the network in the vicinity of x_1 is well connected, few or no flooding requests may be generated. The mechanism of flooding denial described above does not solve the general congestion control problem, but it is essential to a more comprehensive scheme. We also need a mechanism that can force neighbors to reroute a portion of their traffic.

Forced Rerouting

Traffic through x_1, which arrives via its immediate neighbors, can be decreased by rerouting some subset of that traffic. To avoid loops, x_1 must first find a route that makes progress for that subset of traffic. Assume that neighbor x_2 sends packets to x_1, and that x_1 chooses to reroute the subset of traffic associated with x_2 and some destination address *d*. Router x_1 initiates a limited flooding request which it sends only to x_2. We differentiate these *congestion-control* flooding packets from other flooding packets, because these packets contain no higher level data portion to be delivered to some destination. A flooding terminator formats its flooding reply and then drops the packet. Messages that attempt to control congestion or routing must be processed with higher priority than normally transmitted material.

If any flooding replies are generated, they return via x_2. Assume that a reply is received, containing a source route $\{x_1, x_2, \dots, x_n\}$. By consulting its routing table, x_2 will deduce that x_1 is a cul-de-sac for the subset of destinations for which x_n makes progress. This subset includes *d*. The method of eliminating cul-de-sacs discussed earlier implies that x_2 enters the source route $\{x_2, \dots, x_n\}$ into its routing table, and so decreases the traffic load on x_1. Router x_2 may receive several congestion-control flooding replies. As with normal flooding, a choice must be made; but the initial reply received is adequate as the first choice.

The eventual new routing entry and the old entry selected by x_n are marked as having been associated with congestion-control flooding. The old entry is removed and saved for later restoration if x_1 reports that congestion has eased. Router x_1 discards the replies it receives. If x_1 remains congested, then x_1 can choose to perform this procedure again with another neighbor or a differing destination subset.

The destination address *d* used in flooding, when coupled with the neighbor chosen, determines the subset of traffic to be rerouted. What choices should a router make? If one particular neighbor sends the majority of traffic to x_1, that neighbor should be chosen first. It is relatively easy for a router to associate a time-averaged received-packets/second figure with each neighbor. The choice of destination *d* is more complex. We present one mechanism for choosing *d*.

Router x_1 has a routing table with several entries. Each entry specifies a route $[x_1 \rightarrow \dots \rightarrow x_n]$, of length ≥ 1 hop, terminated by some x_n. Router x_1 makes progress toward a subset of destination locations defined by x_n. That subset is the half-plane containing x_n, formed by the line that bisects the chord between x_1 and x_n (see Figure 9). Assume that x_1 chooses destination $d = x_n$ in a congestion-control flooding request. If the flooding is successful, x_2 will receive a source route that makes progress for some portion of the subset and that also avoids x_1. This procedure defines at least one subset for each entry in x_1's routing table. Naturally, if x_2 is to initially receive the congestion-control request, x_1 cannot choose a subset defined by $d = x_2$.

The proper choice for *d* is one that decreases x_1's traffic, but that does not make delivery of rerouted traffic impossible. For a certain subset of traffic, x_1 may be the

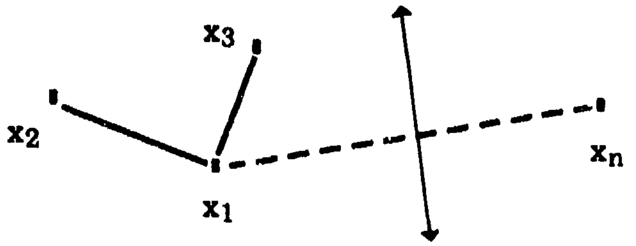


Fig. 9 Rerouting subset selection

only route available. Unfortunately, x_1 cannot know if such a subset exists without complete and current topological knowledge of the internetwork. It may infer that it has selected such a subset if no flooding replies are received. It cannot reroute this traffic and must select another destination.

We assumed that congestion was caused by more traffic arriving at a router than it can realistically service. The router may still have excess bandwidth on its communications links, but it may have too little computational speed to process all the packets it has received. Another sort of congestion occurs when too much traffic is routed over a particular outgoing link *l*. Some degree of buffering is usually available to recover from short periods of excess traffic. However, if the period is sufficiently long, transmission queues for *l* fill, and packets must be dropped.

If the excess traffic is received from neighboring routers, those routers must be made to decrease the amount of traffic they route via *l*. A router can easily accomplish this by sending its neighbors congestion-control flooding packets (other than the neighbor at the other end of *l*). The destinations chosen for the packets must be ones that use *l*. Choosing *d* to be the address of the router at the other end of *l* is sufficient.

Interaction with Flooding Denial

Assume that a neighbor x_2 has decided to invoke flooding denial prior to x_1's congestion. Assume that x_1 now decides to reroute a portion of its traffic via x_2. Now x_2 drops the flooding packet, and as a result x_1 remains congested. If x_2 denies x_1's congestion-control flooding request, x_1 should be informed immediately rather than waiting for the flooding time-out interval to expire. If the congestion-control flooding initiator's immediate neighbor denies that request, it must send a *congestion-control denial* to the initiator. This allows x_1 to rapidly choose another neighbor as a candidate for rerouting. This idea may be extended as follows: If all of x_2's neighbors deny flooding, this implies that x_2 in effect also denies flooding, and so on. It is unclear that this extension is necessary.

Subsequent to accepting x_1's congestion-control request, x_2 may itself become congested. If x_2 were now to send a congestion-control request to x_1, that request would be immediately denied (unless x_1 had become uncongested). To avoid the possibility of x_1 and x_2 rapidly sending congestion-control requests between one another, a short period should elapse after a router becomes uncongested before it again accepts flooding requests. At this time, the router should inform its immediate neighbors that congestion has eased. A *congestion-release* packet sent from x_1 to x_2 provides sufficient information for x_2 to remove from its routing table any entries created by x_1's congestion-control flooding procedure. This procedure causes the associated subset of destinations to be routed back to x_1.

We have explicity constructed a procedure that avoids sending any traffic rerouted as a result of congestion into already congested routers. The interaction of congestion-control flooding and flooding denial allows routes to be constructed that avoid congested regions of a network. The maximum diameter of a congested region that can be avoided is directly related to the flooding limit.

Flooding denial implicitly assumes that sending additional traffic into a congested router diminishes its ability to route traffic. If router x_1 is congested and its neighbors are themselves congested, then any congestion-control flooding initiated by x_1 will immediately provoke congestion-control denial. However, we observe that flooding denial ensures that routers at the outer boundaries of a congested region will attempt to redirect traffic away from the region. In many cases, this will decongest the outer portions of the region, eventually allowing x_1 to decongest.

Applying Cartesian Routing to a Metropolitan Region

We now apply Cartesian routing to the prototypical metropolitan area. Routers in this internet may be connected to one another by any relatively regular tessellation pattern. Without loss of generality, we assume that routers are interconnected by point-to-point links forming a rough mesh. On the average, each router is connected to four neighbor routers. The interrouter distances g_x and g_y have small standard deviation. In the prototype, the network diameter, measured by neighborhood-to-neighborhood router hops, is at least as large as 200.

We cover the metropolitan area with a mesh of routers, most connected only to a few immediate neighbors. A fraction of these are connected additionally to routers quite some distance away. A fraction of those are connected to routers still farther away. This provides a hierarchy organized by connection distance. Each higher level of that hierarchy is composed of routers spaced farther apart. The neighborhood routers comprise the lowest level, (0), in a hierarchy. Normally, level (0) routers also function as gateways. Above this level are the level (1) routers, above them are the level (2) routers, and so on.

Determining Level (1) Router Spacing

It was assumed earlier that the standard deviation from the mean level (0) router-to-router distance is small. Routers were also assumed to be connected into a rough mesh. This allows treating the x and y components of the distance d between source and destination separately. Call those distances d_x and d_y, respectively. Call the mean router-to-router distance in the y-direction g_y, and in the x-direction g_x. The expected number of hops between a source and destination separated by distance d is $d_x/g_x + d_y/g_y$.

Consider a series of routers that are equally spaced and linearly connected. If all source/destination pairs are considered equally likely, the mean value for d is one-third the end-to-end length. Assume for argument that the mesh is laid roughly along compass lines. Let the north-south limit of distance in the internet be D_y hops and the east-west limit D_x hops. The mean of expected hops between source and destination is $D_x/3 + D_y/3$. If $D_y = D_x = 200$, there will be an average of 132 hops between source and destination.

Consider again the linear series of routers. It can be shown that one low-cost method for decreasing hop-count between source and destination over a distance D is to place short-cuts approximately every $i(D/3)^{1/2}$ hops, for $i = 1,2,\ldots,(3D)^{1/2}$ (see Appendix). Level (0) of the internet is composed of a connected mesh, with mean distance $\mathbf{g}_x$ and $\mathbf{g}_y$ between routers. Level (1) may be considered a mesh of 'long-distance' routers separated by distances $\mathbf{g}_x(D_x/3)^{1/2}$ and $\mathbf{g}_y(D_y/3)^{1/2}$. Figure 10 illustrates the hierarchy for a section of such an internet.

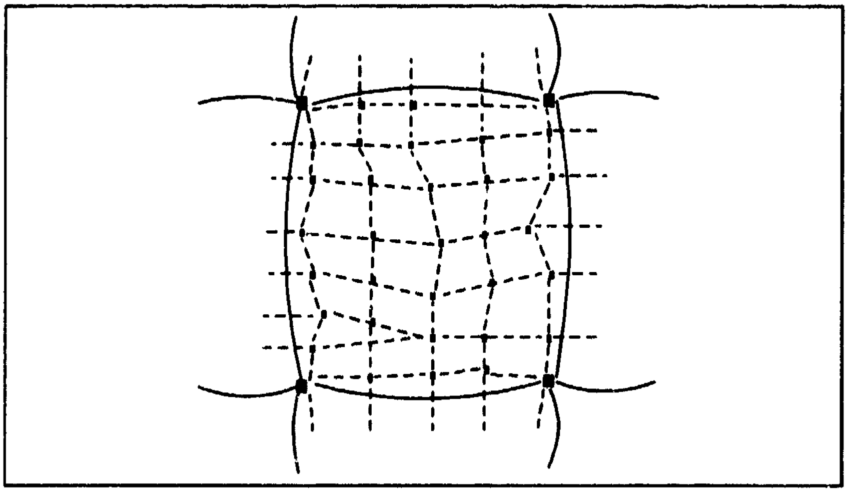


Fig. 10 Portion of level (0) and level (1) connection grids

The prototypical metropolitan area is a square 400 blocks wide. We assume that the neighborhood routers are two blocks apart, so $D_x = D_y = 200$, and $\mathbf{g}_x = \mathbf{g}_y = 2$ blocks. For an internet of that scale, this implies $(D_x/3)^{1/2} = 8\mathbf{g}_x$ distance between level (1) routers, in the x or y direction. From the point of view of a level (0) router, a level (1) router is rarely more than several hops away. Normally, level (0) routers reside in clusters formed by the interconnection of level (1) routers in the four corners.

By repeated application, a level (2) hierarchy can be created from routers spaced approximately $\mathbf{g}_x(D_x/3)^{3/4}$ distance apart. What results is a mesh of level (0) routers spaced one nominal hop, $\mathbf{g}_x$ distance, apart. A level (1) mesh resides above level (0) with a granularity of eight hops between its routers. Finally, a level (2) mesh of routers of granularity 24 hops sits above this. Figure 11 depicts a section of such a hierarchy. The circles represent level (0) clusters bordered by their corner level (1) routers.

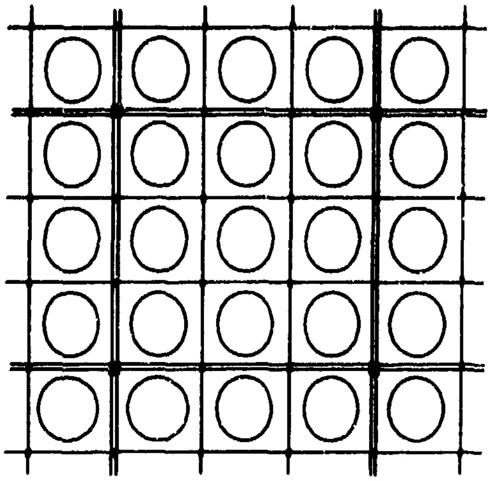


Fig. 11 Sections of three-level hierarchy

Average and Worst-Case Path Length

Under ideal static conditions, any level(1) router is within two hops of its closest level(2) router. Any level(0) router is within eight hops of its nearest level(1) router. Thus ten hops is the static worst-case length of a path from level(0) to level(2). The average distances are one hop and four hops, respectively, and the average static path length from level(0) to level(2) is five hops.

The average internet path length is $2D_x/3 = 132$ nominal hops. In the worst case, $2\cdot[(2\cdot8)+8]=48$ hops are spanned in moving from level(0) to level(2) at the path start, and vice versa at its end. The remaining 84 hops are spanned by level(2) jumps, comprising 24 hops each. This implies a worst-case hop count of $2\cdot10+4=24$ for the average length path. The average hop count for the average length path is $2\cdot5+4=14$. This compares to a figure of 132 level(0) hops if no hierarchy is used.

The longest static internet path length is 398 level(0) hops. Using the hierarchy, this is reduced to 39 hops in the worst case and 29 hops on the average. If the router-to-router packet processing time is kept under five milliseconds, the maximum end-to-end transmission time is under 0.2 seconds, with an average under 0.15 seconds.

Transitions Between Levels

Nodes in level(i), $i>0$, should have links to nodes within their own level and normally to their immediate neighbors in level$(i-1)$. Nodes within a level(i) cluster, $i\geq0$, must maintain source routes to nearby active nodes in level$(i+1)$, assuming that this level exists. These source routes act as a second routing table. Access to this up-level routing table could be triggered by the distance to the destination being farther than some set limiting distance (see Figure 12). Another possibility is to search this up-level table in addition to the normal routing table to determine toward which node to forward a packet. This would approximately double the routing decision time for packets.

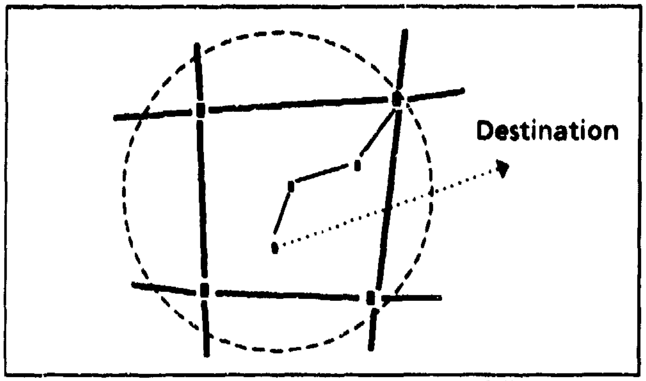


Fig. 12 Transition: level (i) to (i + 1)

A router in level (i), $i>0$, will usually have immediate neighbors in level $(i-1)$ as well as in its own level. A decision to route downward from level (i) to level $(i-1)$ should occur when active neighboring level (i) routers are farther from the destination than the current router. The intention is to find the closest level (i) router bordering the level $(i-1)$ cluster within which the destination resides. This is accomplished by initially treating all immediate neighbors equivalently, placing them into the routing table, and searching for the entry that makes the best progress.

Choosing an Up-level Router

In a multi-level network, the choice of a successor node sometimes requires crossing to a higher network level. This occurs when the distance to a packet's destination exceeds a limit or when no reply has been received to a limited flooding procedure. Normally, a set exists of two or more possible higher level successors. As above, the successor chosen should be that which makes the most progress.

Assume that A is a level (i) router and that it has decided to send a packet up-level. It is possible that no higher level successor makes progress. In that event, the successor chosen should be closest to the packet's destination. This choice violates the rule that all path segments traversed must make progress. The source route to the level $(i+1)$ node may cause that packet to be routed away from its destination in certain topologies. If that level $(i+1)$ node now decides to route back down to level (i), will this cause a routing loop? If it is forbidden to send a packet down-level except via a path segment which makes progress, then loops cannot occur. The Progress Limit Field in each packet keeps track of the closest approach (so far) to the destination. No router will choose to forward to A, because that cannot make progress for the packet. Similarly, no response from a limited flooding procedure could result in a source route which terminates at A. Therefore, no routing loops can occur as a result of level-to-level transitions.

Finally, consider a router in level (i) attempting to forward a packet which has detected a outage of sufficient diameter that limited flooding has failed. The packet's destination is entered into the router's outage table. That packet and subsequent ones for the same destination are then sent to a level $(i+1)$ router as a further delivery attempt. Level $(i+1)$ routers have a longer average hop-to-hop link distance than do those in level (i). Sending packets there under these circumstances will normally provoke limited flooding. Any limited flooding that occurs should span a larger physical area in level ($i+1$) than it does in level (i). This improves the chances for delivery in the case where the initial outage was the result of a large topological irregularity. If there is no higher level than (i), then the destination is assumed to be unreachable.

Topological Irregularities and Router Placement

In response to detection of a topological outage, the source route is modified to use an up-level router. Expected path length can be shortened by judicious placement of up-level routers. Consider Figure 13, which depicts a cluster of level (0) routers in a valley.

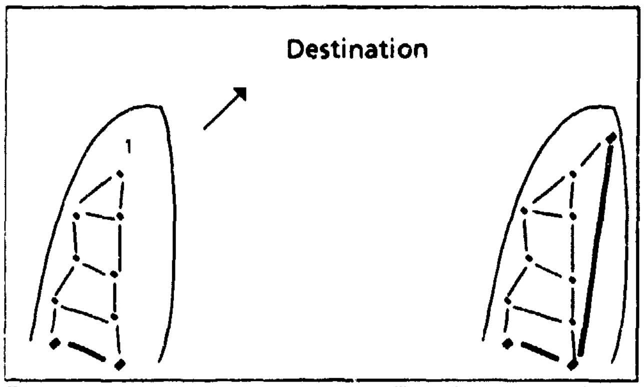


Fig. 13 Routing in a valley

Level (0) routers routing toward the destination send packets to router 1, which detects a topological outage and uses source routing back down the valley to a level (1) router. If an additional level (1) router is placed at the upper end of the valley, several hops are removed from the expected path length. Where practical, level (i) clusters should be 'contained' by level ($i+1$) routers.

Some Practical Design Guidelines

No mention has been made of the data rates, expected load, and traffic patterns needed for evaluating the feasibility and construction of a metropolitan internetwork. No data concerning behavior of users and their network requirements has been collected, because no such internetwork has yet been built. The products and services such an internetwork would provide are a subject of speculation. Any estimates therefore are crude and error-prone.

Current Usage on a Typical Local Area Network

An examination of a current local area network seems the best way at this time to estimate the needed capacity of a level (0) router. We assume that the prototypical local area network has 100 hosts, and that nearly all traffic enters or leaves via the attached router. A survey was taken on a 10 megabit Ethernet containing approximately 100 hosts. The network surveyed is used for personal mail, word processing, database access, and administration. The table below summarizes the results.

<u>Typical Rates During One Hour of Prime Shift</u>

$244.13\cdot 10^3$ packets/hour
$10.53\cdot 10^6$ octets/hour ⇒ 43 octet average packet size
66.28 packets/second for 100 hosts
⇒ 0.66 packets/host-second

Level-0 Router-to-Router Link

Figure 14 depicts a typical level (0) router. Traffic on a level (0) link comes from two sources: packets injected by the local network associated with the router, and packets using the link as an intermediary on their way to the destination. To calculate the load limit on a typical level (0) link, we make the following assumptions:

1. Cluster traffic is uniform on all links.
2. Each router contributes equivalently toward traffic totals.
3. Traffic entering from outside or leaving the cluster is ignored.
4. The cluster is interconnected in a square *x,y*-grid.

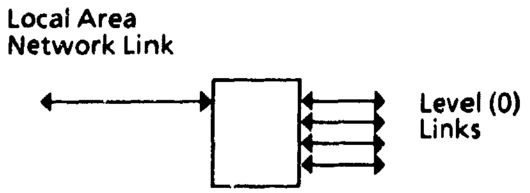


Fig. 14 Typical level-0 gateway

These assumptions allow a rough estimate to be made of traffic on level (0) links by summing traffic sent by all cluster routers and dividing by the number of links. The ratio of hosts to links is $N^2/2\cdot N(N\text{-}1)\sim 1/2$, for large N, where N is the width and height of the cluster. One would normally expect one-quarter of the associated local network traffic to leave via each of the four links. Traffic in a grid using a link as an intermediary is roughly equivalent to traffic generated locally. Thus a level (0) router must be capable of dealing with at least twice the traffic generated by its attached local area network.

Level (1) Routers and Above

To speculate on the traffic patterns used as design guidelines for network levels above level (0) is almost fruitless: very little can be said. The characteristics of Cartesian routing tend to spread traffic across the grid. From the investigation in the Appendix, and assuming traffic to be uniform, links in level (i), $i>0$, should expect traffic load along the x- or y-axis to increase proportional to T(j)/T(1):

$$T(j) = 2\left(Pj - j^2\right),\ j = 1, \ldots, (P-1)$$

where P is the number of level (i) routers in a linear run across the internet, T(1) is the traffic on the first link in the run, and j is the j^{th} link in the run. This assumes that all destinations along the run are equally likely.

Using the prototypical network, where there would be 25 level (1) links, $P=25$ and the ratio $T(j)/T(1)$ for $j=12$ is 6.5. Unless traffic crossing level (1) links is kept low, the quadratic increase in traffic toward the center will limit use. From this it is evident that services whose use generates large amounts of traffic should be kept close to the level (0) clusters that use them, thus greatly decreasing traffic between level (1) routers. One obvious candidate service for this restriction is file servers.

Host Mobility

Host mobility is an important feature of any internetwork, especially one that must be commercially practical. Advances in telecommunications technologies now allow network hosts to be mobile. The most common example of this is a mobile telephone system. Extending packet-switching technology so that it can address mobile hosts is also an essential requirement for a tactical network. In a tactical environment, many hosts are mobile; planes, ships, and trucks are examples. Early work in extending packet-switching technology to mobile hosts was done in the context of the ARPA-Internet by creating new protocols for the transmission of packets over radio [Binder][Frank][Kahn].

The ARPA-Internet protocols were not designed with the ability to access mobile hosts as an objective, nor were they designed to allow indefinite growth. An underlying assumption in the configuration of the ARPANET was that hosts and gateways are fixed. Another assumption was that the destination network is the best path to any of its hosts. Thus the Internet routes toward a destination's network, and not toward its location. As a result, host mobility requires the creation of networks with special protocols which reside as *components* within the ARPA-Internet.

It is time to rethink these assumptions. Host mobility is now allowed only within specialized networks. This is a reasonable approach when mobility is limited to a tiny fraction of hosts (less than one percent). As this percentage rises, it becomes increasingly attractive to incorporate mobility into the internetwork itself. If this is not done, a profusion of specialized mobile networks is created. In the resulting patchwork, mobile hosts may not travel across network boundaries. As specialized networks which incorporate host mobility grow, there will be pressure to create specialized new internetworks for them. In fact, this has already occurred: the SURAN (Survivable Radio Network) project is currently examining issues of scale for packet radio networks [Garcia-Luna][Sacham]. The systems studied are based on the tree-structured routing hierarchy originally developed by Kamoun and Kleinrock [Kleinrock].

We require that the internetwork allow mobile hosts both to retain their connections and to allow access by other mobile and fixed hosts. Host movement should ideally be unrestricted by the internetwork. A Cartesian internetwork that allows host mobility must have routers that can communicate with mobile hosts in sufficient numbers and distributed in such a manner that satisfactory reliability can be achieved with a small flooding limit. This would allow the mobile host subnetwork to grow uniformly, without problems of scale. Associated with this requirement is the ability of an internetwork router to be mobile.

Mobility creates four new classes of routing nodes:

1. The fixed gateway, which supports unattached mobile hosts.
2. The fixed router, which supports unattached mobile hosts.
3. A mobile router that only routes packets. It has no attached network or host population.
4. A mobile router that routes packets and can have an attached host population.

It is probable that some form of packet radio transmission would be used to communicate with mobile hosts and routers. The details of transmission mechanisms to or from mobile nodes of the internetwork system are outside the scope of this report, as are specific details of the protocols required. The precise mechanisms by which mobile hosts exchange information with nearby routers is not discussed, nor how host tables are maintained. This report presents a general framework suggesting how it is possible to incorporate mobile hosts and routers into a Cartesian routing hierarchy. How to efficiently communicate within a cluster of mobile hosts and routers has been a subject studied extensively elsewhere.

We distinguish between attached and unattached mobile hosts as follows: Attached mobile hosts must remain associated with their network, as in a packet radio network. The details of routing inside a packet radio network are not of concern here. An unattached mobile host is free to associate itself with any routers with which it can communicate. It resembles a mobile telephone in this regard. Assuming a population of routers that can communicate with mobile hosts, unattached mobile hosts may be incorporated into a Cartesian internetwork.

In what follows, distinctions are usually not made between a gateway and a router. Gateways are routers with attached networks. Routers are not necessarily attached to specific customer networks. With appropriate equipment, either may be in direct communication with mobile hosts and possibly mobile routers.

Gateways, Address Locations, and Maintaining Connections

Earlier it was suggested that the separation of an address into distinct *location* and *identity* parts would aid in the implementation of host mobility. This form of address allows adoption of a technique that improves reliability for hosts in networks possessing more than one gateway. A natural extension of this technique provides the general host mobility desired.

From the perspective of the ARPA-Internet, any gateway is equally adequate for both entry into and exit from a network. In a Cartesian internetwork, the exit gateway closest to the host is preferred. For networks with only one active gateway, both are equivalent. If a network has more than a single active gateway, then in both systems the sender has a choice of gateways. In a Cartesian system the packet's address inherits its *location* from the gateway used to inject it into the internetwork. Cartesian internetwork *location* assignment is automatic.

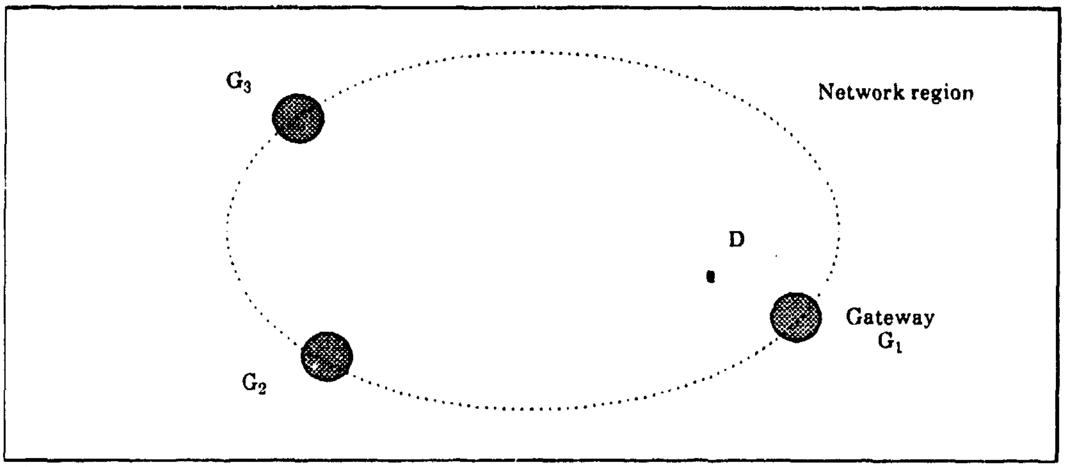


Fig. 15 Network with several gateways

Figure 15 shows a component network with more than one gateway within a Cartesian internetwork. Assume that the gateways are widely separated and that host D has an open connection. Gateway G_1 is the preferred gateway, since it is closest to D. Packets leaving D and entering the internetwork via G_1 inherit their *location* from G_1. Subject to the uniqueness restriction, the *id* is whatever the network prefers. Packets returning to D will use G_1's *location*. What happens if G_1 ceases to operate?

When a neighbor encounters the inoperative link to G_1 and a packet arrives for G_1's *location*, limited flooding attempts to find some closer router. If the limited flooding fails, the packet will then be dropped and the connection appears broken. Shortest-path routing has an advantage here, since it would soon recover and reroute via either G_2 or G_3. But this is only half the problem, since D must also begin to use one of the other two gateways.

If only the link from G_1 out to the internetwork has failed, G_1 can institute the necessary local recovery procedures. An ICMP-style redirect message is sufficient to effect recovery [Postel]. This solution works only if G_1 is operating. As a practical matter, D should notice that G_1 is not operating after some interval or be informed by the network that it should no longer use G_1. This is a problem for both types of internetwork. This event should cause D to send a duplicate packet immediately, and subsequent packets via either G_2 or G_3.

If these facilities function, then the connection need not be dropped in a Cartesian internetwork. For this form of recovery to operate, there must be two or more gateways on the affected network. Assume that G_1 has failed and the network has begun sending D's packets via either G_2 or G_3. Packets from D inherit the *location* portion of their Cartesian address from the exiting gateway. Its *location* changes to either G_2 or G_3, but its *id* remains the same. A connection can be maintained if the other end uses the most recently received *location* information from D.

When a connection is opened, the source uses a standard known *location* for the destination, presumably that of a principal gateway. The source obtains this information via a local table or enquiry to a domain name service [Mockapetris]. Throughout the duration of the connection, the source monitors the *location* portion of the destination addresses in packets it receives from the destination. The source will receive at least acknowledgment packets from the destination. In all packets it sends to the destination, it uses the most recently received *location* information when building the destination address. This allows packets to reach hosts via alternate gateways for certain classes of failure. This situation is strikingly similar to a mobile host moving between routing nodes.

Cartesian Routing and Mobile Hosts

Mobile hosts present a severe problem for traditional network routing algorithms. A large part of the problem is a lack of location information concerning the whereabouts of a mobile host at a particular time. This is exacerbated by forms of address which do not include location information. For example, an ARPA-Internet address specifies only a destination network whose location may span a continent.

Consider a mobile host B, traveling past routing nodes of known location as in Figure 16. If we assume that B is in communication with at least one routing node, then its position can be estimated as 'close' to that routing node. The Cartesian address of any routing node with which B is in direct contact provides sufficient information to route packets addressed to B to that node and thence to B.

A *mobile-host-address* is of variable length but is limited by the number of routing nodes considered realistic. We denote a mobile-host-address by the bracketed list $<r_1,r_2,...,r_n,id>$, where $r_1,r_2,...,r_n$ are the routing nodes and *id* contains the mobile host's unique identifier. Suitable network addresses for the mobile host are the set of Cartesian addresses $[r_1,id]$, $[r_2,id]$, ... , $[r_n,id]$. The *routing-node-list* could be supplied in packets sent from a mobile host by including them in a packet header option similar to that used in the ARPA-Internet protocol [RFC 791].

Maintaining Connections to Mobile Hosts

Cartesian routing sends packets toward destination locations. Assume that every packet sent from a mobile host contains its current mobile-host-address, both its *routing-node-list* and *id* portions. Any host receiving packets from a mobile host has the location of the routing node(s) with which the mobile host was recently in communication. The sender uses the most recent address it has received from a host for the destination address when building packets. As long as the period between packet exchanges is kept sufficiently short, packets sent to the mobile host follow it.

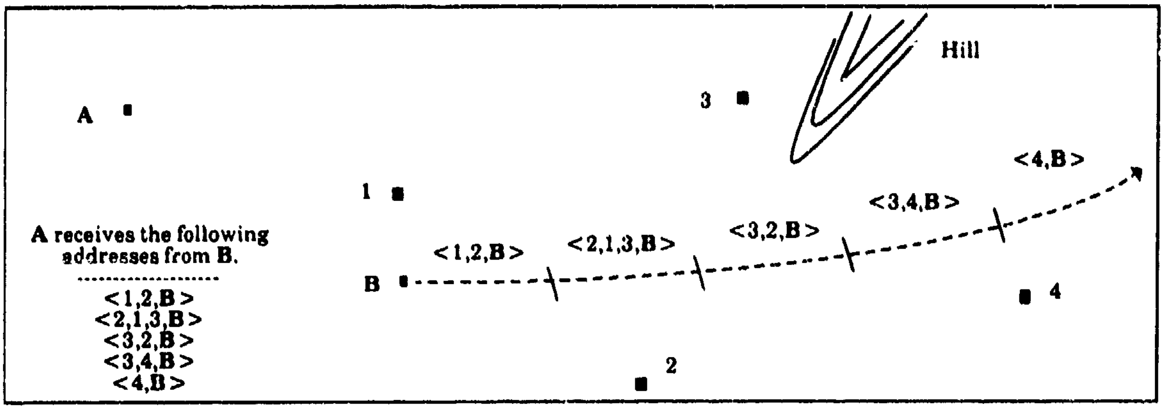


Fig. 16 Mobile host addresses

This assumes that a path does exist to the destination with an outage diameter no greater than the flooding limit.

To maintain a connection, a mobile host must remain in contact with at least one routing node. At any point a mobile host may be in contact with more than one node. Thus a sender to a mobile host may have some choice of destination routing node when building packets. To help make this choice, the mobile host should report in its mobile-host-address the routing node locations in order of reliability. This may mean reporting the routing node with the strongest signal first. The sender would normally choose the most reliable routing node as the location portion of the destination mobile host's address. If communication proceeds poorly or if the traffic is of an urgent nature, each packet sent to the mobile host could be copied repeatedly. Each packet would contain a different routing node from the mobile-host-address. This maximizes probability of delivery, with the expense of duplication.

We see how this works in Figure 16. Host **A** is in communication with mobile host **B**. Initially, **A** uses the address [1,B], which is the most reliable routing node from the mobile host address <1,2,B>, supplied by **B**. In succession, **B** replies with new addresses as it moves, <2,1,3,B>, <3,2,B>, etc. Finally, it sends <4,B> as it passes behind the hill and loses communication with routing node 3. **A** uses addresses [2,B], [3,B], ..., and finally [4,B], as they are received from **B**.

It is clear that both ends of a connection may be mobile. Packets sent by either host follow their respective destinations on their paths. Both hosts must use the most recent mobile-host-address and their own current mobile-host-address when building packets.

The property of choosing a primary routing node can be used to the advantage of both parties. (We assume that the mobile host knows its own location and heading. The routing nodes with which it is in communication have already supplied the mobile host with their locations). If a mobile host is moving away from one routing node and toward another, then presumably the routing node it is approaching is the best destination to use in the near future. The mobile host can take advantage of this by sending acknowledgments which contain its predicted best mobile-host-address. The time between sending this prediction and using the new address is realistically bound from below by the average round-trip time.

From the point of view of an intermediate router, packets addressed to a host at a fixed location are indistinguishable from those addressed to a mobile host. This is an important feature of Cartesian routing. If an internetwork transport tier uses Cartesian routing, mobile hosts may move transparently, without breaking connections, as long as contact is maintained with some non-partitioned router. This is a feature not generally available in other routing procedures.

Maintaining the Quality of Estimated Position

A mobile host provides an estimate of destination position by means of the locations of nearby routing nodes. This estimated position is used by a source communicating with a mobile host. The quality of an estimate depends upon two factors: the velocity of the mobile host and the time elapsed since the source received a packet from the mobile host. The two factors are related: the greater the mobile host velocity, the more frequently should the source receive updated destination addresses, thus maintaining the accuracy of the source's estimate of destination location. This implies that a mobile host should ensure a minimum interval between sending packets. The interval period is inversely related to its velocity. If the period expires, then a repeated acknowledgment packet which reports the current mobile-host-address should be sent to the source.

Obtaining Initial Router Locations

For a connection with a mobile host to be initiated, the location of the routers in contact with it must be known. If routers maintain a cache of mobile hosts with which they are in direct communication, then an inquiry arriving at any router can provoke a reply specifying whether or not the router is in contact with a particular mobile host. Reaching the necessary set of routers could be achieved by flooding, or by an *indefinite enquiry* (see below).

It would be more efficient to maintain a network service, similar to a domain server, which would list mobile hosts and the routers in contact with them. Routers should update their service entries whenever they lose contact or come into contact with a mobile host. For the purposes of initiating a connection, the initial set of routers in recent contact with a desired mobile host may then be determined by enquiry. To allow such a mechanism to scale upwards indefinitely would require the use of some hierarchy in the *id* portion of an address.

Indefinite Enquiry

In certain situations, such as the existence of mobile hosts, mentioned above, the precise address of a host may not be known. In such a case, we assume that the host desiring to open a connection has a good guess about the approximate destination location. More precisely, the source host knows the destination host *id* and its location to within some distance, Δd, of a known location (x,y). See Figure 17.

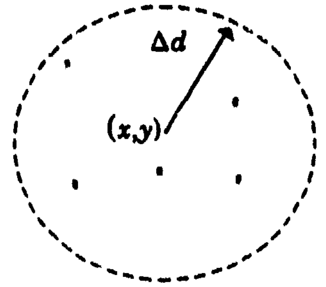


Fig. 17 Indefinite enquiry

The source sends out an *indefinite enquiry*, specifying the search center location (x,y), the search radius Δd, and the host address [$*$,*id*]. The asterisk denotes an unspecified *location*. Routing nodes in the network know their own *id* and those of the hosts with which they are in direct communication. Cartesian routing tries to send the enquiry packet close to position (x,y). Any router which receives the packet, within Δd distance of (x,y), initiates a variant of limited flooding called *enquiry flooding*. Copies of the packet are sent to any neighbor within the circumscribed region. A reply is sent from any router in direct communication with a host of matching *id*. The reply, [(x_i,y_i),*id*], contains that router's current location (x_i,y_i) and the original *id*. Not all routers within the search area necessarily receive an enquiry flooding packet. If a reply is expected but not forthcoming, then the radius Δd can be increased and the enquiry reissued, or the search center (x,y) can be altered.

Mobile hosts are presumably in contact with some routing nodes in their vicinity. An indefinite enquiry provides a mechanism by which a focused search for a mobile host may proceed. Closely allied is the problem of determining which hosts are inside a certain area. This is accomplished by a similar mechanism, in which the host address is completely unspecified, as in [$*$,$*$]. Routing nodes within the search area respond with their addresses and those of their hosts. For any indefinite enquiry, the source host must be prepared to accept many replies.

Mobile Routers

Until now, routers have had fixed positions. Only hosts could be mobile. We now discuss issues that arise when a router itself may be mobile. There are several reasons for providing mobile routers. A mobile router may be used to reconnect a partitioned portion of the network or to provide redundancy, thus preventing partitioning. It may be temporarily positioned to provide a shorter route between network regions or adding capacity. A group of hosts in an aircraft or ship residing in a local area network would have a mobile gateway.

For Cartesian routing conditions to be met, the neighboring routers, with which the mobile router can communicate, must know the mobile router's approximate position. Conversely, the mobile router must know the approximate positions of its neighboring routers. These conditions are similar to those for a mobile host. Each router maintains a routing table containing the locations of its neighboring routers. Cartesian routing efficiency is affected by the accuracy of the location information in that table. If a neighboring router is mobile, its location in the routing table must periodically be updated. If the location information for a mobile router is sufficiently out-of-date, that router is assumed no longer to be a neighbor.

It is the responsibility of mobile routers to periodically exchange current addresses with neighboring routers (mobile or fixed) that have the ability to communicate with it. As a result, these neighboring routers update their routing tables. If the routing table contains no entry matching the *id* in the address, a new entry is created, containing the address and a time stamp. (This occurs when a mobile router moves into the reception area of a new router.) If the routing table does contain an entry with matching *id*, the *location* portion of that entry is updated and the time stamp reset. Periodically, any routing table containing such entries is examined for those whose time stamps are older than some preset interval. Entries that are older are removed from the table, and it is assumed they are no longer reachable. A broadcast-like mechanism must be used by a mobile router to 'discover' the set of routers with which it can communicate. To ensure a viable two-way path between a mobile router and any neighbor in this set, a simple three-way handshake with a short time-out interval can be used to exchange addresses.

Effects of Mobility on Unstable Routing Loops

In our earlier discussion of routing loops, only host mobility was considered; therefore, inaccuracy in estimated host position did not affect routing decisions in a way that could cause a routing loop, because the host was an end-point of communication. However, once a router is allowed to be mobile, inaccuracies in its estimated position can cause routing loops. Stable routing loops can be avoided as long as mobile routing nodes periodically exchange addresses and update their routing tables. Unstable routing loops remain possible, their duration controlled by the frequency with which mobile routers exchange addresses with their neighbors. There may be short periods when inconsistent routing data is used and unstable routing loops occur. A mechanism to prevent these short-duration loops is desirable.

A mobile router presumably knows its current location more accurately than any of it neighboring routers. If it uses its current location for the purposes of routing determination, then routing loops can develop. In Figure 18, assume that mobile router M was at its old position when it exchanged addresses with B. B chooses M for the next hop upon receipt of a packet destined for D, because M is closer than C. However, M is actually at its new position, and B's decision is based on obsolete data. Upon receipt, M routes using its new location and forwards to B, thus causing the loop.

Any routers in communication with a mobile router have recently exchanged addresses. B therefore has an estimate of M's position. Since M is mobile, B broadcasts packets to it. B must identify M by address as the intended recipient for the packet, since other routers may intercept the transmission. Assume that the physical link protocol for this type of communication encapsulates the data with sending and intended receiving router addresses. When B transmits a packet to M, this provides B's current estimate of M's location (also an updated estimate of B's

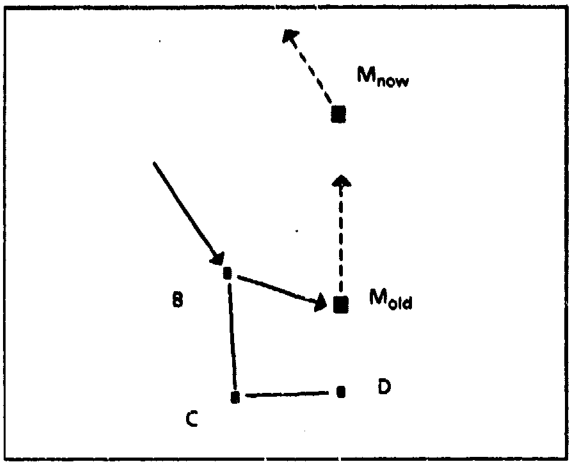


Fig. 18 Unstable routing loops

location). If M determines that B is closer to the destination and forwards the packet back to B, it also provides B with its current location, thus breaking the loop.

Mobility: Some Other Issues

We now discuss ancillary issues that arise when incorporating mobile hosts. One question is: How does a router know if a packet has reached its destination? If a router is in *direct communication* with the destination, then that router believes it can contact the destination without the use of intermediary routing nodes. A gateway to the destination's local area network is an example. A packet radio router with strong, direct line-of-sight communication to the destination is another.

Identifying Destinations

It is usually simple for a fixed router to determine whether or not it needs to forward a packet farther through the internetwork. If the *location* portion of the destination address matches that of the router's, the packet is not forwarded and the destination host is assumed to be in direct communication with the router. Mobile hosts complicate this solution, since their location changes.

A *mobile-host-address* specifies a destination router *location* in each packet header. The routing proceeds to a destination location as before. If a fixed router can be in direct communication with mobile hosts, it must have a host table containing the *id*'s of all mobile hosts with which it believes it is currently in direct communication. It matches the destination *id* of a packet against that table. If a match is not found, it follows recovery procedures or discards the packet.

If mobile routers can be in direct communication with mobile hosts, the situation is more complex. Mobile routers have a varying *location*. The *location* portion of a *mobile-host-address* can only be presumed to be 'close' to the destination, since a mobile router is presumably near any mobile host with which it was recently in contact. The mobile router cannot in general determine from the *location* part of a *mobile-host-address* whether or not it should continue to forward the packet. It must always check the destination *id* against a host table.

Mobility and Limited Flooding

Limited flooding, initiated by a router that has no neighbors closer to a packet's destination, causes a controlled search for a router closer to the destination than the initiator. If the search is successful, the terminator sends a reply to the initiator, containing a source route between them. For routers accessing mobile hosts, the terminating condition must be altered.

A router that participates in limited flooding receives from the initiator the distance from the initiator to the destination and the destination address. The participating router then calculates the distance between itself and the destination. It successfully terminates if that distance is less than the distance it received from the initiator. To adapt to the addition of mobile hosts, the distance between a router and a mobile host is defined as *zero* if and only if the router believes it is in *direct communication* with the destination. If it is not in direct communication, the router calculates distance in the normal manner. For routing purposes, the host is treated as if it were attached to the routing node.

The initiator receives an improved source route to the destination from the set of routers that successfully terminated the flooding procedure. If this set contains a reply from a router in direct communication with the destination, the distance to the destination in at least one reply will be zero. For mobile destinations, such replies are preferred over non-zero replies with shorter hop counts. This reflects a bias which assumes that the mobile subnetwork has a best path to the destination. If more than one reply with a zero distance is received, choice is left to the initiator. The header of the terminator's reply message may provide the initiator with information that may affect its choice (such as signal strength in a packet radio subnetwork). In many cases, however, the closest zero-distance terminator will be used. In the event that no zero-distance replies are received, normal procedures are followed.

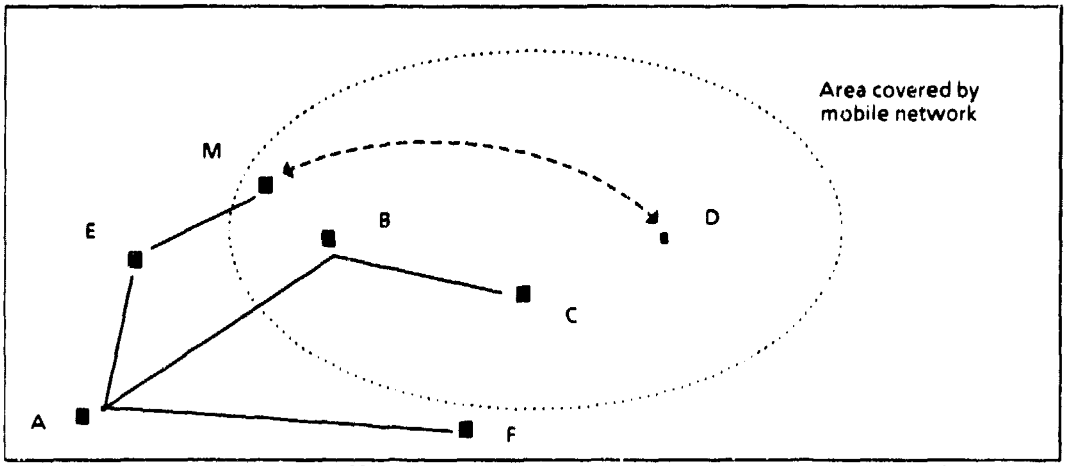


Fig. 19 Effect of mobile hosts, zero distance, and limited flooding

Host and router mobility alter somewhat the behavior of limited flooding. The preference for zero distance replies means that routes closer to the destination location are discriminated against in favor of those in direct communication with the destination. This is illustrated in Figure 19. Assume that M and D are in a mobile subnetwork, M is in direct communication with D, and F has a packet intended for D. Since it cannot route closer to the destination, F initiates limited flooding. The source route from F to M is preferred over closer and shorter routes via B or C, because M reports a zero distance while B and C do not.

A mobile router r_j which is part of a source route $\{r_1, ..., r_i, r_j, r_k ...\}$ is expected to lose communication with either r_i or r_k more frequently than a fixed router. This creates an outage along the source route. As with fixed routers, when r_i notices that it can no longer communicate with r_j, the source route is used as a reverse path to alert r_1 that its current outage table entry for that destination is invalid and a new source route may be needed. Possibly, r_1 will have stored alternate source routes from its previous limited flooding result. Otherwise it must reinitiate limited flooding. It is wise to include a provision for mobile routers to indicate their mobility in limited flooding responses. All else being equal, a source route without mobile routers is usually preferable to one with mobile routers.

Simulating Cartesian Routing

Figure 20 is a simplified flow diagram of a particular implementation of multi-level Cartesian routing. We will briefly discuss some aspects in the application of Cartesian routing in preparation for simulating its performance on an existing network topology.

Choosing a Closer Neighbor

The Cartesian routing algorithm allows choosing any successor node which makes progress toward the packet's destination. The candidate successor nodes form a set. When more than one node is in the set, a question arises: Which node should be chosen? The best choice requires complete topological knowledge, but this is not available. Simulation suggests adoption of the greedy principle: choose the successor node that makes the best progress toward the destination.

Choosing an Up-level Limit

In large networks there are often hierarchies of point-to-point links with increasingly larger point-to-point distances. A network level from 0 to N is associated with each level of hierarchy. Level (0) contains clusters of nodes with the shortest average link lengths. Level (N) contains clusters with the longest average lengths. The up-level limit is a distance measurement. If a node within network level (i) receives a packet whose destination is farther away than this limit, the node assumes that some higher network level can more efficiently deliver the packet. The correct choice of the up-level limit depends upon local topology. It should be no closer than the closest node in network level ($i+1$).

Simulation of Cartesian Routing in the ARPANET

The development of Cartesian routing is motivated by three observations: it scales upward indefinitely, it is robust, and it enables mobile hosts to be addressed in

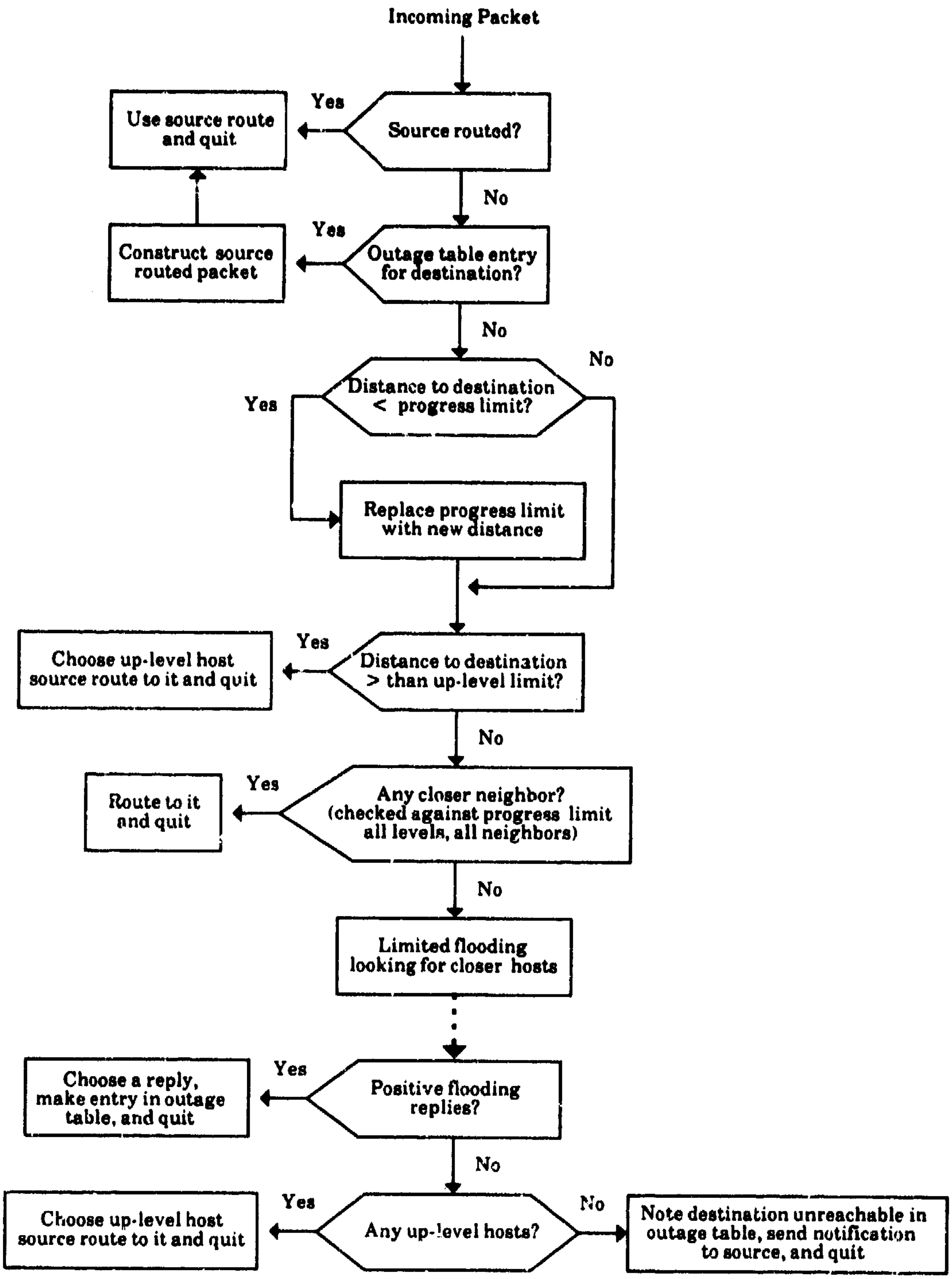

Incoming Packet
Source routed?
Yes
Use source route and quit
No
Outage table entry for destination?
Yes
Construct source routed packet
No
Distance to destination < progress limit?
Yes
No
Replace progress limit with new distance
Distance to destination > than up-level limit?
Yes
Choose up-level host source route to it and quit
No
Any closer neighbor? (checked against progress limit all levels, all neighbors)
Yes
Route to it and quit
No
Limited flooding looking for closer hosts
Positive flooding replies?
Yes
Choose a reply, make entry in outage table, and quit
No
Any up-level hosts?
Yes
Choose up-level host source route to it and quit
No
Note destination unreachable in outage table, send notification to source, and quit

Fig. 20

Multi-level Cartesian routing

a nearly transparent manner. However, its behavior has yet to be demonstrated. A simulator was written in Interlisp to allow comparison of one- and two-level Cartesian routing with shortest path routing in a known network.

The ARPANET was the obvious choice for this comparison. The ARPANET topology was obtained from a January 1986 geographic map, and approximate latitude and longitude locations were assigned to each of the 48 IMP nodes. The route taken by each of the 2256 source/destination paths was determined for one- and two-level Cartesian routing, and shortest-path routing algorithms.

It is important to remember that the connection pattern of ARPANET node-to-node links is highly irregular. There is no attempt in the simulation to impose any measure of Cartesian regularity to the pattern. It is not the pattern a network administrator would choose if it were known that Cartesian routing was to be used. By slightly redefining network connectivity, much better results can be obtained.

Cluster Definition

A cluster is a related group of interconnected network nodes. If no hierarchy is to be employed, all the nodes of the network fall within a single level (0) cluster. If a multi-level hierarchy is employed for reasons of efficiency, it is the task of the network administrator to define the clusters at each level of the network. In this case, clusters within the same level should be composed of nodes for which the standard deviation of the node-to-node link lengths is small. If a network is small or sufficiently irregular, no hierarchy should be employed.

Proper cluster selection increases the routing efficiency of the network. Cartesian routing performs best in networks of regular and repeating patterns of interconnection. In realistic, densely populated networks, cluster selection will be easy. In a sparse and irregularly populated network, the task is more difficult. The ARPANET contains four obvious level (0) clusters: Central California, Southern California, New England, and Washington, D.C. Figure 21 shows the level (0) Southern California cluster in the ARPANET. ISI22, UCLA, and USC possess links that are many times longer than the other links in the cluster. They are the obvious candidates for inclusion in level (1) of a two-level hierarchy.

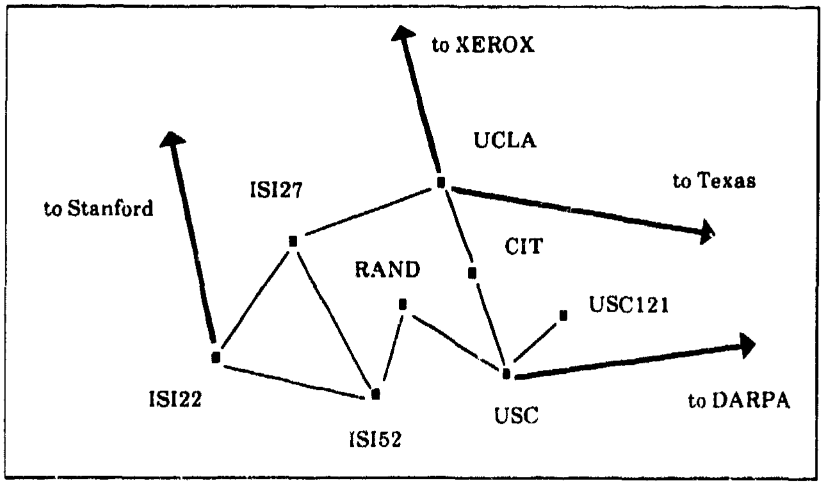


Fig. 21 ARPANET Southern California cluster

Choosing the Set of Up-Level Nodes

In a sufficiently irregular network not designed with Cartesian routing in mind, nodes in a level (i) cluster also connected to longer distance links may not be good choices for the cluster's level ($i+1$) neighbors. (Figure 8 demonstrates this.)

Normally, the choices for up-level nodes for the Southern California cluster would be {ISI22, UCLA, USC}. UCLA is the northernmost node of the cluster. However, because UCLA has a link to Texas, it is on the best path to the southeastern region of the ARPANET. Similarly, although USC is the southeasternmost node in the cluster, it is on the best path to the northeastern region.

When routing up-level, Cartesian routing normally chooses the nearby up-level node closest to the destination. In this situation, that procedure produces poor routes. This can be greatly alleviated by choosing the end-points of long-distance links as the cluster's level $(i+1)$ neighbors. A better set of up-level nodes for the Southern California cluster is {DARPA, Stanford, Texas, XEROX}.

The added expense is an increase in level (i) node's routing table. The source routes to nearby level $(i+1)$ neighbors are longer, and there may be more of them. However, a substantial improvement in routing can result from a more judicious choice of these neighbors. Algorithms to automatically determine the correct choices are possible.

Simulation Results

Graph 1 shows shortest-path routing plotted against the simulation results for one- and two-level Cartesian routing. The horizontal axis represents sets of paths of a particular hop-count length, the vertical axis the number of paths in a set. The black bar represents shortest-path routing, the other two (respectively) one- and two-level Cartesian routing. Each algorithm had the same number of single-hop paths. They are excluded from the graph.

The shape of the graph shows that the three algorithms have similar behavior. The average path length, over all paths, is shown below.

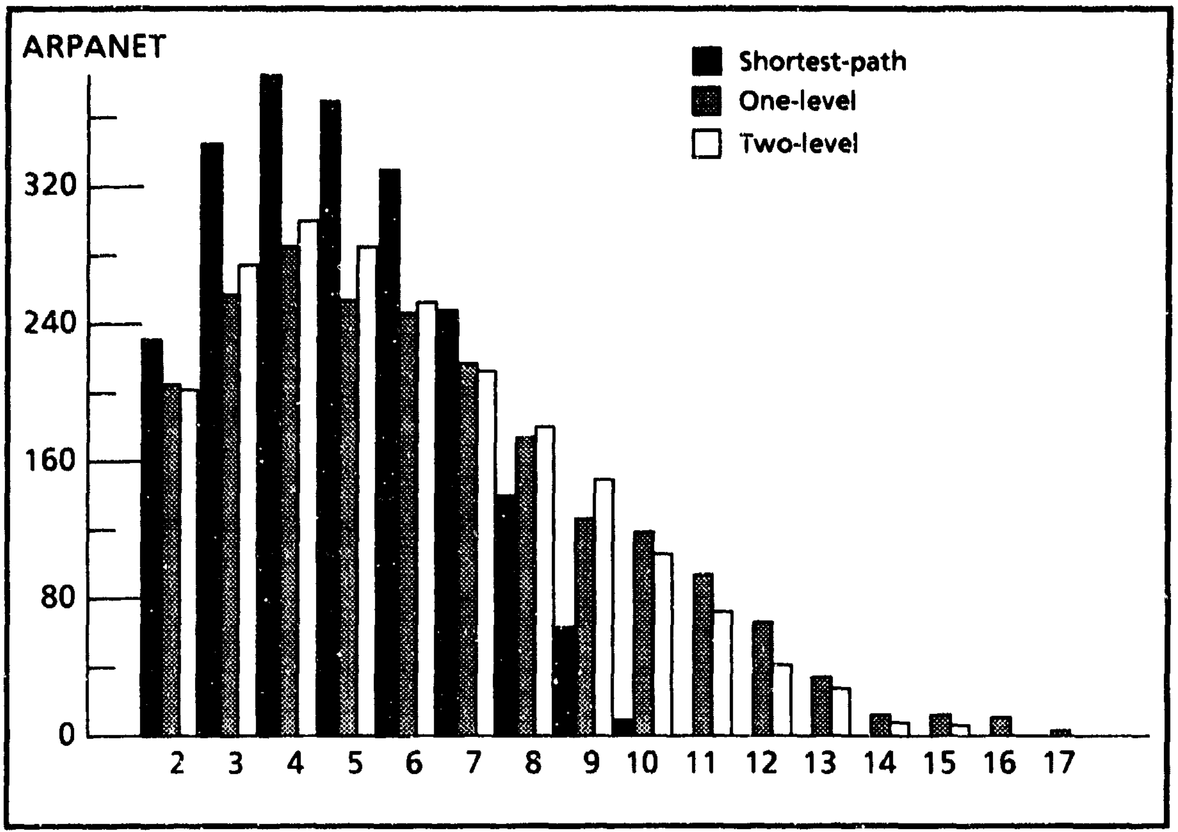
Shortest-Path	One-Level	Two-Level	
4.67	5.93	5.66	
	27%	21%	Percentage increase

For the ARPANET, two-level Cartesian routing produces an average path length increase of 21%. The average path is one hop longer. The longest paths are 10, 17, and 16 hops, respectively, for shortest-path, one-level, and two-level Cartesian routing. Two-level routing has only one path of length 16.

We list the average and longest source route lengths seen in the simulation:

One-Level	Two-Level	
3.4	3.5	Average source route length
5	6	Longest

Cartesian routing applied to the ARPANET demonstrates that it can operate efficiently with an average increase in path length of 21%. It does so while keeping much smaller routing tables, and without the need for any topological information other than that in a node's immediate vicinity. Finally, as the size of a network



Graph 1 Shortest path vs. one- and two-level Cartesian routing

expands and connection patterns become more regular, the difference in path length between shortest-path routing and multi-level Cartesian routing will decrease.

Conclusions

Our routing algorithm has been shown to be suitable for internetworks of unbounded size. It responds rapidly to outages of limited diameter by constructing source routes around the outage. Traffic may be categorized into differing levels of importance, with greater efforts made to deliver more important traffic. The algorithm allows nearly transparent host mobility by separating location from identity in the address. Mobile hosts may move freely between component networks within the internet, as long as they remain in communication with some routing node.

Each router within a hierarchic internetwork level keeps a minimum of information; a router need only know the locations of its near neighbors and source routes to routers one level above it. End-to-end delay is kept small by building a multi-level hierarchy, each level possessing links of greatly increased length. Robustness is achieved by carefully limited flooding, initiated by the node that discovers the outage, which searches for a route that makes continued progress possible. Unlike the ARPA-Internet routing algorithm, this cost is paid only when an outage is encountered, and only by that portion of the internetwork in proximity to the outage. By assuming increasing regularity for higher internetwork levels, it is possible to keep low the probability of an outage that partitions source from destination.

Appendix

Minimizing Hop Counts in Point-to-Point Networks

Assume a simple point-to-point network with a length of several kilometers. Assume further that the number of attached nodes is large and that each node is spliced into a cable which connects all the nodes. The topology may be that of a ring or a linear segment. Consider laying another cable parallel to the first. This Appendix concerns itself with the question of how to utilize bounded amounts of additional cable (or channels within a single cable) to minimize the expected hop count for the average packet travel path.

The Model

Consider a point-to-point network segment formed when connecting a series of P nodes to one another with a cable. Assume that transmission time is small with respect to node processing time, and that all destinations are equally likely. As P becomes large, the average number of hops and hence the time required to send a packet from source to destination becomes excessive. Several feasible mechanisms considered for reducing this delay were ruled out. CSMA technology, for example, is inefficient over the distances considered. Satellite technology has a long round-trip time and is prohibitively expensive.

This appendix examines a solution that utilizes bounded amounts of additional cable to provide short-cuts spanning several hops. It is important to point out that the addition of *physical* cable to achieve short-cuts is of course unnecessary. The reservation of specific communication channels within a single cable can function logically as additional cables. This is attractive where substantial extra bandwidth is likely to be available, as with a fiber-optic cable.

If nodal processing time is large with respect to transmission time, then decreasing end-to-end delay is synonymous with decreasing the average hop count. We could reduce the average hop count to one merely by connecting each node to every other node. However, for networks with a large number of nodes or long internode distances, that is unrealistic. Such a solution requires a prohibitively large amount of cable to be laid between nodes: an amount which is cubic with respect to the average internode distance. Perhaps the next most reasonable method for lowering the hop count would be to connect the nodes to form a balanced binary tree. This would lower the hop count to $\log_2(P) + 1$, but it is exponential in required cable length.

It is clear that increasingly better solutions can be found by employing increasingly larger amounts of cable. We will begin by restricting our solutions to twice the amount of original cable. If a second cable is laid alongside the first, it becomes possible to connect nodes to one another to form *short-cuts*. Figure A1 shows the topological description of one such network with these short cuts. It is interesting to ask the following questions: How much improvement in average hop count do we obtain from such a pattern of short-cuts, and which patterns result in the most improvement?

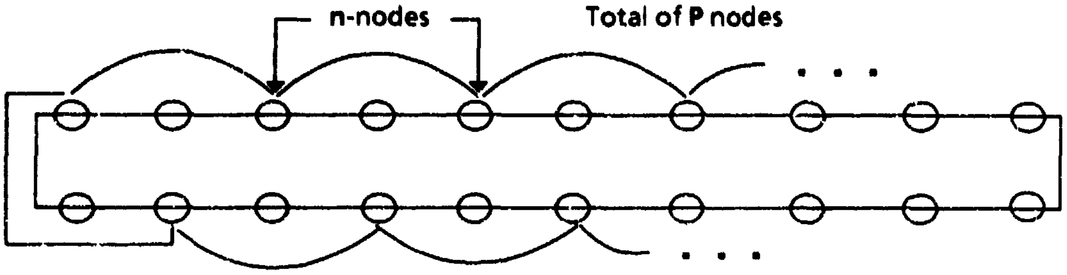


Figure A1 Ring network segment with short-cuts

Ring Topology

Pick any node and number it zero. The first short-cut interconnection pattern which comes to mind connects node 0 to node **n**, node **n** to node **2n**, etc. (for now, we assume that P is evenly divisible by n). This strategy obeys the 2x cable limitation for large P and is amenable to treatment as a continuous system. Given an average path length of l_{av} measured by hop count before the addition of short-cuts, what is the optimal value for **n**?

We assume that all source/destination pairs are equally likely and that internodal distances are roughly equivalent. The average path length is l_{av} and the average path crosses l_{av}/n short-cuts, neglecting error introduced by paths that do not start or stop on short-cut boundaries. To correct for this, approximately n/2 hops are made between the start and the initial short-cut and between the last short-cut and the destination. An expression for the expected hop count is:

$$H_1 = 2\left(\frac{n}{2}\right) + \frac{l_{av} - 2\left(\frac{n}{2}\right)}{n}, \; P \geq 4. \qquad \text{(A1)}$$

By differentiating and solving for the minimum, the optimum value for n is:

$$n_{opt} = \sqrt{l_{av}} \;\; \text{yielding} \;\; H_{1opt} = 2\sqrt{l_{av}} - 1. \qquad \text{(A2)}$$

In Graph A1, l_{av}/H_{1opt} is plotted against P for $l_{av} = P/4$, corresponding to the average path length for a ring. As expected, the improvement grows with P. For P=100 the number of hops is decreased by a factor of three, for P=1200, by nine. Although the value for l_{av} is assumed to be P/4, this is not correct for small values of P. The exact expression for l_{av} rapidly approaches P/4 as P increases.

To apply this to an actual network, we should round the real valued n_{opt} to the nearest integer. It does not matter where we begin placing the short-cuts. If P is not evenly divisible by the rounded n_{opt}, then at the conclusion of placing the short-cuts, some number of nodes may remain unspanned. We span them with a single short-cut.

Reapplication

We can consider the second cable, installed by applying the above algorithm, as comprising a ring network of its own. Conditions that were needed to apply the algorithm originally still hold. The traffic is still uniform and the average path crosses approximately $(l_{av})^{1/2}$ short-cuts. We generalize this. By successively reapplying the algorithm, we obtain connections between nodes:

$$\left\{0, l_{av}^{1/2}, 2l_{av}^{1/2}, 3l_{av}^{1/2}, \dots, \left(l_{av}^{1/2}-1\right)l_{av}^{1/2}, 0\right\} \quad 1^{st}\text{ application.}$$

$$\left\{0, l_{av}^{1/4}l_{av}^{1/2}, 2l_{av}^{1/4}l_{av}^{1/2}, 3l_{av}^{1/4}l_{av}^{1/2}, \dots, \left(l_{av}^{1/4}-1\right)l_{av}^{1/4}l_{av}^{1/2}, 0\right\} \quad 2^{nd}\text{ application, etc.}$$

Each reapplication increases the amount of cable by a factor of one. At the optimum, the average hop count obtained by successive reapplication is:

$$H_{1opt} = 2\,l_{av}^{1/2} - 1$$

$$H_{2opt} = l_{av}^{1/2} + 2\,l_{av}^{1/4} - 1$$

$$H_{kopt} = \sum_{i=1}^{k-1} l_{av}^{1/2^i} + 2\,l_{av}^{1/2^k} - 1, \text{ for } k \geq 2. \qquad \text{(A3)}$$

H_{kopt} rapidly approaches a limit as k grows. Improvement stops when $l_{av}{}^{1/2^i} = 2l_{av}{}^{1/2^{i+1}}$. For practical networks of a few hundred nodes, this limits reapplication to $k = 2$. In Graph A2, H_{kopt} is plotted for a range of k and P.

Linear Topology

The continuous model is most accurate for a ring network where the average traffic density is constant at all points along the ring. What if the ring is broken? Assume instead the linear topology as described by Figure A2. The average path length can be calculated and is $l_{av} = P/3$, for large P, perhaps a non-intuitive result.

The continuous model developed for a ring is not as accurate in this case. The average traffic density can be expected to increase toward the center due to the absence of a path between the two end nodes. Shortest paths which went *around the end* in the ring topology must now cross the center. To account for this, intuition suggests that a better solution would place larger short-cuts toward the center.

As an alternative, what improvement would be made by connecting the first and last nodes? This obeys the 2x cable limitation and makes a ring from the linear network, thus shortening the average path length from $l_{av} = P/3$ to P/4. Any extra cable used for short-cuts now violates the 2x limitation. By avoiding connecting the

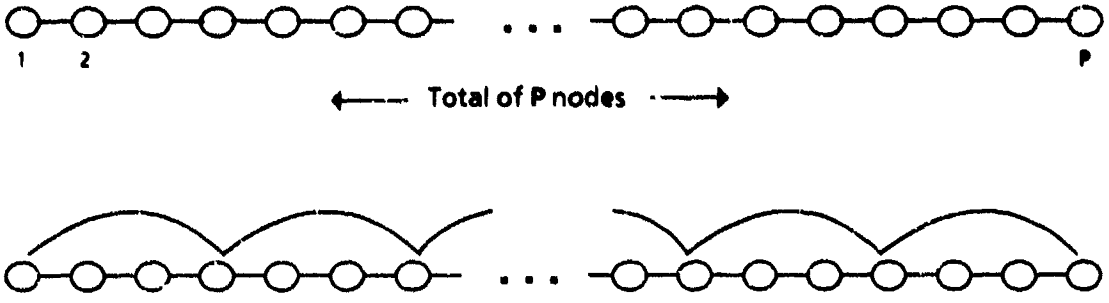


Figure A2 Linear network segment (above) with short-cuts (below)

ends, can a pattern of short-cuts be found which obeys the limitation and yields a better improvement?

Starting at the end, we number the nodes from one. Because the ring has been split, there is one less node-to-node link. Again, assuming all source/destination node pairs to be equally likely, there are 2(P-1) paths which cross link [1←→2]. They are: (1,2),(1,3),...,(1,P), which crosses [1→2], and (P,1),((P-1),1),...,(2,1), which crosses [1←2]. For every path (x,y) there is a matching path (y,x). Using this symmetric argument, the number and length of paths crossing a link from left to right are equal to those crossing from right to left. Consider now a typical shortcut in Figure A3, which connects nodes i and j, where $j \geq i+1$. From the left there are (i) sources for $(P-j+1)$ destinations.

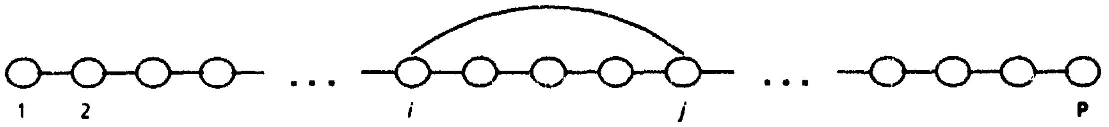


Figure A3 Typical short-cut

For $P \geqq 2$, this results in Equation A4.

$$T(i,j) = 2\Big[\ i(P-j+1)\ \Big], \text{for } P \geq j \geq i+1. \tag{A4}$$

$T(i,i+1)$ predicts the number of paths which cross a particular link and so models link traffic as a function of position. Call this $T(i)$, where link [1←→2] is identified with $i=1$ and link [(P-1)←→P] with $i=(P-1)$.

$$T(i) = 2\left(Pi - i^2\right),\ i=1,\ldots,(P-1). \tag{A5}$$

Graph A3 shows $T(i)$, plotted against link position for a linear network with $P=9$. The link traffic function can be used to determine link capacities. Knowing the used capacity of [1←→2], the required capacity of link [i←→$i+1$] must be $T(i)/T(1)$ larger than [1←→2] or [(P-1)←→P]. The probability that a particular packet will cross a given link is easily calculated. There are (P^2-P) unique paths. $P(i)=T(i)/(P^2-P)$ yields the probability that a path will cross a particular link.

In the ring topology, minimizing the expected hop count for the average-length path is a straightforward optimization decision. Traffic is everywhere uniform, hence equivalent-length shortcuts. Traffic for the linear topology is nonuniform. A different approach should be used.

One approach to minimizing the hop count for the average-length path is to maximize the number of hops saved by the addition of short-cuts. The sum of hops taken by all paths is then lessened, resulting in a shorter average path length. $T(i,j)$ is used to evaluate the amount saved by a particular short-cut. The number of hops saved by a short-cut over all paths is:

$$S(i,j) = T(i,j)(j-i-1),\ \text{for } j \geq i+2. \tag{A6}$$

By calculating the values of $S(i,j)$, given P, for admissible values of i and j, we obtain a triangular [(P-2),(P-2)] matrix. Such a matrix is presented in Figure A4 below, for P=9.

$$\begin{array}{cccccccccc}
i= & [1] & 14 & 24 & 30 & 32 & 30 & 24 & 14 & \\
 & [2] & 0 & 24 & 40 & 48 & 48 & 40 & 24 & \\
 & [3] & 0 & 0 & 30 & 48 & 54 & 48 & 30 & \\
 & [4] & 0 & 0 & 0 & 32 & 48 & 48 & 32 & \\
 & [5] & 0 & 0 & 0 & 0 & 30 & 40 & 30 & \\
 & [6] & 0 & 0 & 0 & 0 & 0 & 24 & 24 & \\
 & [7] & 0 & 0 & 0 & 0 & 0 & 0 & 14 & \\
 & & [3] & [4] & [5] & [6] & [7] & [8] & [9] & =j
\end{array}$$

Figure A4 $S(i,j)$ for P=9

It is straightforward to write an algorithm that maximizes the short-cut sum for a given $S(i,j)$ matrix. This algorithm must generate $S(i,j)$ for all partitions of P where $(j-i) \geq 2$. Execution time grows rapidly with P, and it is inappropriate to execute for the values of P we wish to examine.

The sum of all path lengths may be determined by adding $T(i)$, for i from 1 to P. A closed-form expression for it is:

$$L(P) = \frac{P^3 - P}{3}. \tag{A7}$$

The exhaustive algorithm was run for values of P from 6 to 33. The optimal set of solutions was always found to be composed of contiguous short-cuts. Within the optimal set, at least one element was composed of a short-cut pattern that started at node one and finished at node P.

A Continuous Model of the Linear Network

The difficulty in finding the optimum short-cut pattern for large P suggests an analytic approach based on a continuous model. To make the system of equations tractable, we convert the formula for $S(i,j)$ into a form incorporating the shortcut length.

$$S(i,i+x) = 2i(P-i-x+1)(x-1),\ \text{for } x \geq 2. \tag{A8}$$

Experience gained from the exhaustive algorithm suggests that the model concern itself only with contiguous short-cuts, as in Figure A5. The initial short-cut begins at node 1 and ends at node $(1+x_1)$; the second short-cut begins at $(1+x_1)$ and ends at $(1+x_1+x_2)$; etc. This determines a system with one variable for each short-cut.

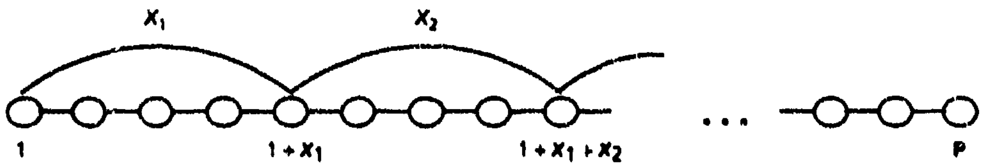


Figure A5 Contiguous short-cuts

If $S_P(\cdot):R \rightarrow R$ represents the savings from a system of one short-cut, with P nodes, beginning at node 1 and with domain x a real, we generalize and $S_{(P,N)}(\cdot):R^N \rightarrow R$ represents a system of N short-cuts, with the domain of x a vector $<x_1, x_2, \ldots, x_N>$.

The formula for $S_{(P,N)}$ is the sum of N applications of $S(i,i+x)$, once for each short-cut:

$$S(1,1+x_1)=2\,(P-x_1)(x_1-1).$$

$$S(1+x_1+\ldots+x_{i-1},1+x_1+\ldots+x_i)=2\left(1+\sum_{k=1}^{i-1} x_k\right)\left(P-\sum_{k=1}^{i} x_k\right)(x_i-1),\text{ for } 2\leq i\leq N. \qquad \text{(A9)}$$

The resulting system is:

$$S_{(P,N)}(<x_1,\ldots,x_N>)=\sum_{i=1}^{N} S(1+x_1+\ldots+x_{i-1},1+x_1+\ldots+x_i). \qquad \text{(A10)}$$

with the restrictions:

$$x_1,\ldots,x_N \geq 2 \text{ and } 1+x_1+\ldots+x_N \leq P.$$

To be meaningful, no short-cut can have length less than two; this places an upper limit on the number of allowable short-cuts (P-1)/2. To determine which pattern of short-cuts yields a nearly optimal pattern for a given P requires that $S_{(P,j)}$ be maximized for each j. It also must be determined which j yields the overall maximum, $j=2,\ldots,(P\text{-}1)/2$.

Each system for $j\geq 3$ superposes j-1 cubic terms and one quadratic term. They are not, in general, unimodal. We expect many local maxima. This system was presented to the unconstrained nonlinear minimization algorithms of Shanno and Phua [Shanno] and More and Cosnard [More].

The behavior of the algorithms was not as good as the author had hoped, confirming the suspicion of many local maxima. Shanno and Phua's algorithm allowed the user explicit control over the search region for each x_j and was found to be the most useful. The algorithm was run iteratively over j for each P, and P was varied over the range of 33-512. Equipartitioning the distance (P-1) among the short-cuts seemed to produce a local maximum at each $j=2,\dots,(P-1)/2$. The algorithm identified the number of short-cuts j at which $max\ S_{(P,j)}$ was largest.

Equipartitioning presumably represents one of many local maxima for $S_{(P,j)}$. It is not a global maximum, and better solutions almost certainly exist where the x_i are properly constrained. This suggests an approach to the integer valued problem for large P. A short-cut length $\lfloor (P-1)/j \rfloor$, for optimum j as determined by the algorithm, should produce a solution that approximates the ideal solution in the continuous system. Intuition suggests that the remainder be absorbed by placing short-cuts of length $\lceil (P-1)/j \rceil$ toward the center. By analogy with Equation A1, this produces an average expected hop count of:

$$H=\frac{P-1}{j}+\frac{j}{3}-1. \qquad \text{(A11)}$$

The decision to equipartition brings us back to the simpler model for the ring network. Differentiating and solving for the minimum,

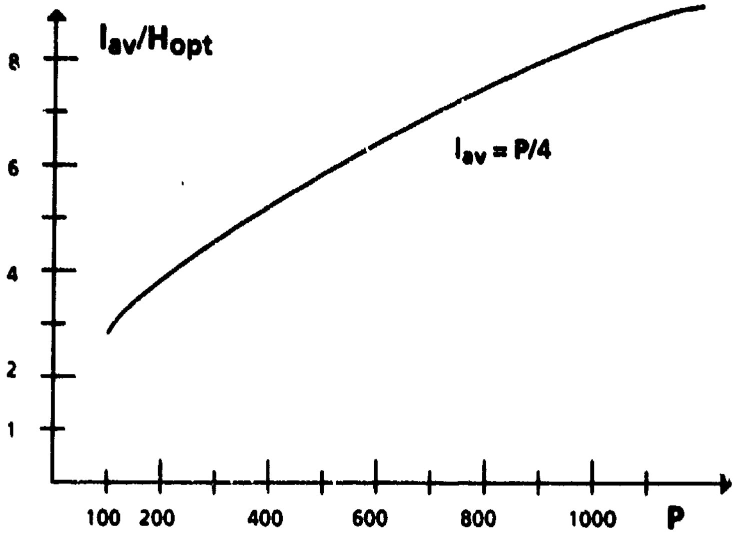
$$j=\sqrt{3(P-1)}.$$

To check whether the assumption concerning the integer solution was true, the suggested algorithm was programmed. For an integer equipartitioning at the optimum number of short-cuts j, the ratio of total path length without short-cuts to total path length with short-cuts is displayed in Graph A4. Graph A5 displays a curve approximating the optimal j, as a function of P. Over the range examined, this curve agrees closely with the optimal j derived from Equation A11. It can be seen that, for linear point-to-point networks, the addition of a second cable allows us to decrease the average hop count by factors of 2.4 to 7.1 over the range of nodes 33 to 512, respectively.

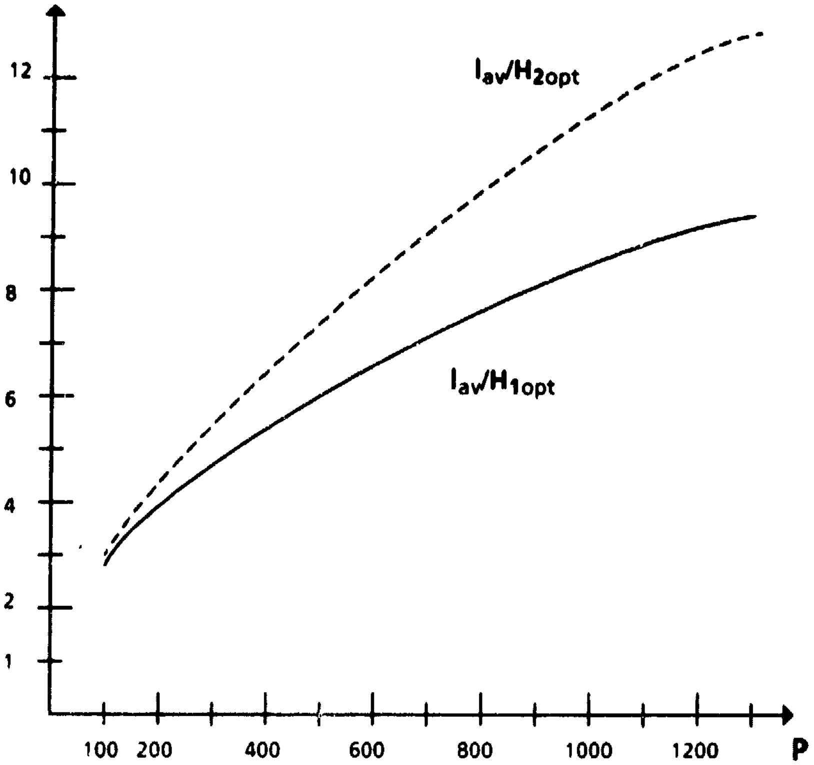
Of interest is the difference between the integer equipartitioned result and the true integer optimum. This is known only for P up to 33. For P from 16 to 33, the difference between integer equipartitioning and the optimal integer solution did not exceed 5 percent. It seems safe to conclude that equipartitioning provides a nearly optimal result.

Extension by Composition

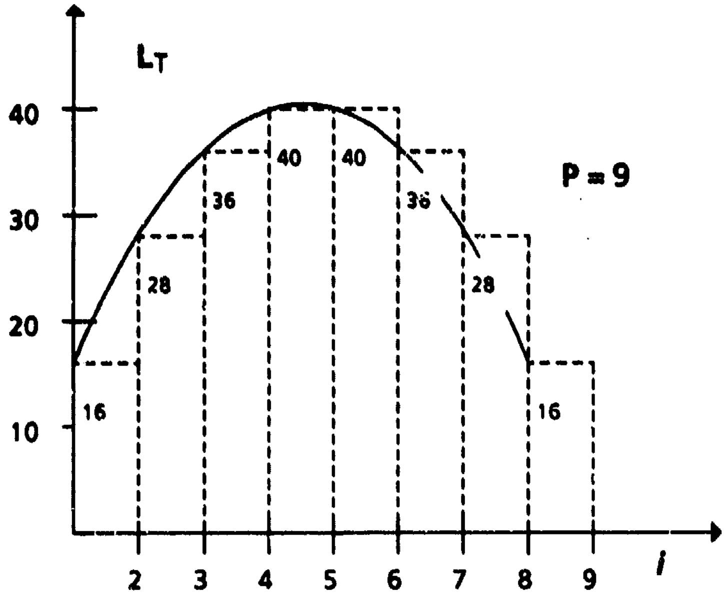
An obvious objection to these results is the restricted topology imposed on the network. It is difficult to construct 'interesting' networks when restricted to a single cable. However, more complex topologies may be constructed by composition. A grid network is composed by laying a series of parallel linear network segments, which are interconnected by laying another series across the former. By applying optimization to each linear segment independently, the average hop count for the grid is reduced. Other topologies are of course possible.



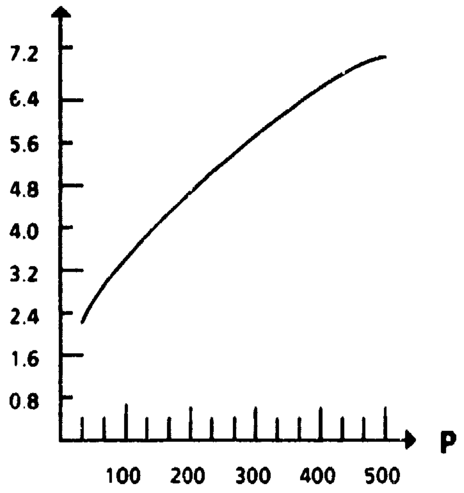
Graph A1 I_{av}/H_{opt} vs. P



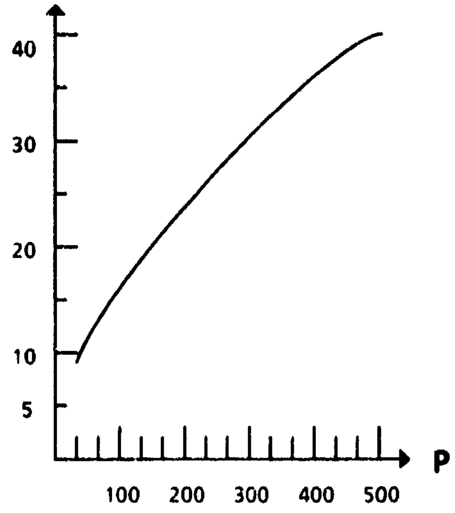
Graph A2 I_{av}/H_{1opt} and I_{av}/H_{2opt} vs. P



Graph A3 Link traffic vs. link position



Graph A4 Equipartition Improvement



Graph A5 Optimal *j* vs. P

References

[Binder] Binder, R., Abramson, N., Kuo, F., Okinaka, A., Wax, D.
ALOHA Packet Broadcasting - A Retrospect
AFIPS Conf. Proc., Vol. 44, pp. 203-215, 1975.

[Burchfiel] Burchfiel, J., Westcott, J., Lauer, G.
Clustering Algorithms for Adaptive Hierarchical
Organization of Large Networks
Bolt Beranek and Newman, Inc.
Report No. 5427, SRNTN No. 5, October 1983.

[Cerf] Cerf, V.G., Kahn, R.A.
A Protocol for Packet Network Interconnection
IEEE Transactions on Communications
Vol. COM-22, pp.637-641, May 1974.

[Cerf-2] Cerf, V.G., Kirstein, P.T.
Issues in Packet Network Interconnection
Proceedings of the IEEE
Vol. 66. pp.1386-1408, November 1978.

[Dijkstra] Dijkstra, E.W.
A Note on Two Problems in Connection with Graphs
Numerische Matematik 1:269 (1959).

[Ford] Ford, L.R., Jr., Fulkerson, D.R.
Flows in Networks
Princeton University Press, 1962.

[Frank] Frank, H., Gitman, I., Van Slyke, R.
Packet Radio System - Network Considerations
AFIPS Conf. Proc., Vol. 44, pp. 217-231, 1975.

[Garcia-Luna] Garcia-Luna, J. J., Sacham, N.
Analysis of Routing Strategies for Packet Radio Networks
IEEE Infocom '85, pp. 292-302, May 1985.

[IEN 189] Rosen, E.C.
Issues in Internetting Part 4: Routing
Bolt Beranek and Newman, Inc.
June 1981.

[Jaffe] Jaffe, J.M., Moss, F. H.
A Responsive Distributed Routing
Algorithm for Computer Networks
IEEE Transactions on Communications
Vol. COM-30, pp. 1758-1762. July 1982.

[Kahn]
Kahn, R.E., Gronemeyer, S.A., Burchfiel, J.,
Kunzelman, R.C.
Advances in Packet Radio Technology
Proceedings of the IEEE
Vol. 66, No. 11, November 1978.

[Kleinrock]
Kleinrock, L., Kamoun, F.
Hierarchical Routing for Large Networks
Performance Evaluation and Optimization
Computer Networks
Vol. 1 (1977), pp. 155-174.

[Kleinrock-2]
Kleinrock, L., Kamoun, F.
Optimal Clustering Structures for Hierarchical Topological
Design of Large Computer Networks
Networks
Vol. 10 (1980), pp. 221-248.

[McQuillan]
McQuillan, J.M., Falk, G., Richer, I.
A Review of the Development and
Performance of the ARPANET Routing Algorithm
IEEE Transactions on Communications
Vol. COM-26, pp. 1802-1811, December 1978.

[Merlin]
Merlin, P.M., Segall, A.
A Failsafe Distributed Routing Protocol
IEEE Transactions on Communications
Vol. COM-27, pp. 1280-1287, September 1979.

[More]
More, J.J., Cosnard, M.Y.
BRENTM, A Fortran Subroutine for the Numerical
Solution of Systems of Non-Linear Equations
ACM Transactions on Mathematical Software
Vol. 6, No. 2, pp. 240-251, June 1980.
(See Algorithm 554 in *Collected Algorithms From ACM*.)

[RFC 789]
Rosen, E. C.
RFC 789: Vulnerabilities of Network Control Protocols:
An Example
Bolt, Beranek and Newman, Inc.
September 1981.

[RFC 791]
RFC 791: Internet Protocol
DARPA Internet Protocol Specification
USC/Information Sciences Institute
September 1981.

[RFC 792]
RFC 792: Internet Protocol
DARPA Internet Control Message Protocol Specification

USC/Information Sciences Institute
September 1981.

[RFC 823] RFC 823: The DARPA Internet Gateway
Bolt Beranek and Newman, Inc.
September 1982.

[Sacham] Sacham, N., Tornow, J. D.
Future Directions in Packet Radio Technology
IEEE Infocom '85, pp. 93-98, May 1985.

[Schwartz] Schwartz, M., Stern, T.E.
Routing Techniques Used in Computer
Communications Networks
IEEE Transactions on Communications
Vol. COM-28, No. 4, April 1980.

[Shanno] Shanno, D.F., Phua, K.H.
Minimization of Unconstrained Multivariate Functions
ACM Transactions on Mathematical Software
Vol. 2, No. 1, pp. 87-94, March 1976.
(See Algorithm 500 in *Collected Algorithms From ACM*.)

[Sunshine] Sunshine, C.A.
Interconnection of Computer Networks
Computer Networks
Vol. 1, No. 3, January 1977.

[Xerox] Xerox Corporation
Internet Transport Protocols
XSIS 028112, December 1981.