ELSEVIER SCIENCE B.V. [DTD 4.1.0]
JOURNAL COMPNW ARTICLE No. 2380
PAGES 01-12 DISPATCH 26 June 2000
COMPNW 2380 A
PROD. TYPE: FROM Disk

ELSEVIER

Computer Networks 000 (2000) 000–000

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

# Cartesian routing

Larry Hughes [a,*], Omid Banyasad [b], Evan Hughes [c]

[a] *Department of Electrical and Computer Engineering, Dalhousie University, Halifax, Nova Scotia, Canada B3J 2X4*
[b] *Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada B3J 2X4*
[c] *School of Computer Science, Carleton University, Canada*

Received 13 September 1999; received in revised form 17 April 2000; accepted 25 April 2000

**Abstract**

Cartesian routing is a novel packet routing methodology in which a packet's route is determined by the position of the router relative to that of the destination. Cartesian routing differs from existing provider-based unicast routing in that routing tables are unnecessary since communications are topologically dependent, thereby potentially reducing router and network overheads. For example, routing decisions, which can take O(log(*n*)) to O(*n*) time using routing tables is reduced to O(1) in Cartesian routing. This paper describes Cartesian routing for unicast communications within a local or metropolitan environment (e.g., limited areas, including buildings, suburbs, and small towns that exhibit an anthropogenic ordering) using two-dimensional address structures, such as latitude and longitude. A Cartesian network is constructed from two types of router: collector (for horizontal communications) and arterial (for both 'horizontal' and 'vertical' communications). Cartesian routing requires minimal state: typically, routers need only know their location and the *reachability* of arterial routers on their collector. The paper also shows how the proposed 128-bit IPv6 address structure is an ideal candidate for Cartesian addresses. 

*Keywords:* Routing; Geographic addressing; IPv6

## 1. Introduction

Existing hierarchical routing schemes, employing distance vector routing and link state routing, require the exchange of routing information to construct routing tables [1]. As networks grow in size, the memory requirements for the routing tables and the time taken to search the tables increase proportionally [11]. Further, as the popularity of computer networks increases, the size of the address space can become a limiting factor [7]. Although many Internet routers employ specialized caches to hold recently-used addresses, table search times can degenerate to O(log(*n*)) [1] and O(*n*), for ordered and unordered tables, respectively [2,10].

Cartesian routing overcomes many of the problems associated with hierarchical routing schemes: there are no routing tables, minimal state information is exchanged between routers, route calculation is neither memory nor CPU intensive, and the address space is virtually limitless, given the 128-bit IPv6 address structure [3].

* Corresponding author. Tel.: +1-902-494-3950.
*E-mail address:* larry.hughes@dal.ca (L. Hughes).

[1] Where '*n*' is the number of entries in the routing table.

1389-1286/00/$ - see front matter 
PII: S1389-1286(00)00130-4

The Cartesian routing methodology is based upon two distinct co-operative *linear* routing algorithms and is intended primarily for local or metropolitan communications (i.e., communications within a limited area such as a building, suburb, or small town). These algorithms require each router to maintain minimal state information and are designed to minimize router complexity and the time required to formulate routing decisions. These routers keep, forward, and discard packets by comparing their location (for example, a floor and office number, or a latitude and longitude) with the packet's destination address, expressed in the same units as the router's location; routing tables are neither required nor used. As a result, routing decisions can be made in O(1) time, without the need for specialized hardware.

The remainder of this paper is organized as follows. In Section 2, a linear routing methodology for a one-dimensional topology is described, as it is the basis of both Cartesian routing methodologies. Section 3 examines the Cartesian routing algorithms, in terms of the permissible topologies, addressing, and network interconnection. The steps involved in initializing a Cartesian network are examined in Section 4, while the fault tolerant aspects of the routing algorithms are discussed in Section 5. The Cartesian address structure must be large enough to represent a router's horizontal and vertical location, Section 6 shows the applicability of the IPv6 address structure to Cartesian routing. The paper concludes with a summary of this research and a description of current research activities.

## 2. Linear routing

The fundamental principles of Cartesian routing can be illustrated using a linear routing algorithm in an one-dimensional network topology. In this topology, each router is associated with two ports (east and west), allowing it to connect to, at most, two other routers. Every router is bound to a unique address and maintains no state information other than this address. The routing algorithm, originally described in [5], is the basis of the Commkit Wide Area Network [6].

Linear routing is achieved in the network by imposing an ordering on the routers which is based upon the unique router addresses; for example, west-to-east in ascending order (unless otherwise specified, west is always considered 'less than' east). When a packet is to be transmitted on the network, the transmitting router's layer 3 (Network/Internet layer) determines the packet's initial direction by examining the destination address. If this is less than the router's address, the packet is queued for transmission on the west port, otherwise on the east port:

```
wait_for_pkt(&port, &packet);
if (packet.longitude < router.longitude)
  if (port == EAST)
    forward(WEST, &packet);
  else
    discard(&packet);
else
if (packet.longitude > router.longitude)
  if (port == WEST)
    forward(EAST, &packet);
  else
    discard(&packet);
else
  keep(&packet);
```

The `forward()` function queues the packet on the specified port for subsequent transmission, while the `keep()` function passes the packet to a higher layer for subsequent processing.

`Discard()` can operate in one of two ways: the packet can be discarded without informing the source; or, alternatively, a control packet can be returned to the source, indicating that the intended destination does not exist. If a control packet is used, it is necessary to modify `discard()` so that control packets destined for non-existant destinations are also discarded.

This algorithm maintains minimal state since it is not necessary for a router to 'know' the addresses or locations of any other routers on the network. Should a router be removed, other routers are not informed. The addition of a new router must ensure that the address ordering is main-

tained; it is not necessary to inform other routers of the addition. The constant number of comparisons used, yields a O(1) cost per routing operation.

## 3. Cartesian routing

Linear routing, despite its simplicity, exhibits a number of limitations. For example, packets must visit all routers between the source and destination routers, and short-cutting between non-adjacent routers is not permitted since it would violate the routing algorithm. Furthermore, dividing a single, large linear network into several smaller ones is also problematic, since linear routing does not support communications between distinct linear networks.

These limitations can be overcome by constructing a two-dimensional (Cartesian) network from horizontal and vertical linear networks. To achieve this, three extensions to the linear network topology are necessary:

- A Cartesian topology is composed of two or more horizontal 'subnetworks', or *collectors*, consisting of routers connected horizontally, and sharing a common horizontal identifier, such as latitude. Collectors are interconnected by one or more vertical subnetworks, or *arterials*, consisting of routers connected vertically; arterials need not share a common vertical identifier.
- The address structure must indicate the location of a router in two dimensions; that is, the router's location is specified in terms of its horizontal and vertical positions, such as latitude and longitude.
- There are two types of routers: collector and arterial. A collector router has two horizontal ports, while an arterial router has two horizontal ports and, typically, a minimum of two vertical ports. Each router type has its own routing algorithm, variants of the linear routing algorithm described in Section 2.

Although the Cartesian routing algorithm can apply to any ordered, two-dimensional topology of horizontal and vertical connections (e.g., offices on a floor and floors in a building), the remainder of the paper will assume that collectors run east–west and arterials run north–south in a "geographical" address space.

### 3.1. Topology

The topology of a Cartesian network is dictated by router connectivity and ordering, meaning that:

- There must be at least one path between the most northerly collector and the most southerly; this ensures that there is a route between any two communicating entities on the network.
- A packet must be discarded as soon as a router determines that the packet's destination address does not exist.

How the above requirements are supported is summarized in the following sections.

#### 3.1.1. Basic arterial/collector connectivity

In a Cartesian network, every arterial must intersect with each collector it passes at an arterial router. There are two reasons for this requirement.

First, if a collector fails to connect to an arterial, packets can be discarded incorrectly. In Fig. 1, packets destined for collector router C via router A1 and forwarded to router A2 will always be discarded by router A2, since router C is 'south' of router A2. Similarly, packets for router C sent from router A2 to router A1 will always be dis-

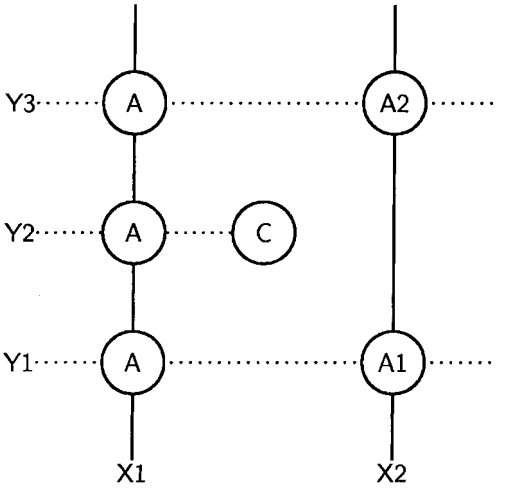


Fig. 1. Collector Y2 without an arterial router on X2 (Collectors are dotted lines, while arterials are solid.)

carded by router A1, since router C is 'north' of router A1. The solution is to ensure that the topology is designed so that each collector has an arterial router for every arterial, as shown in Fig. 2.

The second reason is best illustrated by considering the situation in which one or more arterials terminate without connecting to a collector. In Fig. 3, four arterials terminate without their most northerly routers connecting to a common collector. If a packet destined for router A4 (longitude 0) is sent north on the arterial at longitude −30, when the packet reaches the northernmost router, it will be discarded since there is no collector running east–west. One possible solution is to modify the routing algorithm to allow the packet to be forwarded backwards to an arterial that can forward the packet along a collector. Not only does this require significant changes to the entire routing algorithm, it also adds to network traffic as the packet retraces its route. The solution adopted, illustrated in Fig. 4, is to ensure that all unconnected arterial routers are connected to a common collector.

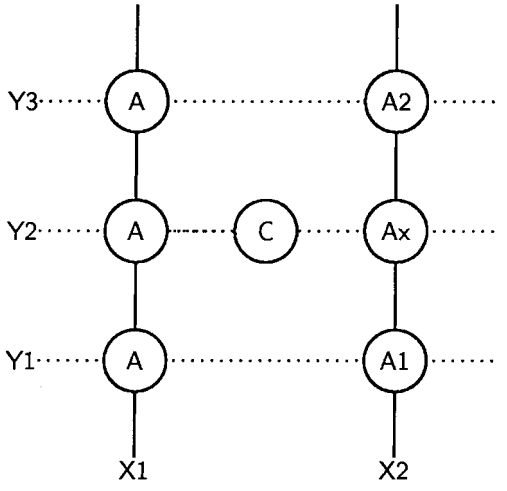


Fig. 2. Collector router C is now reachable via arterial X2.

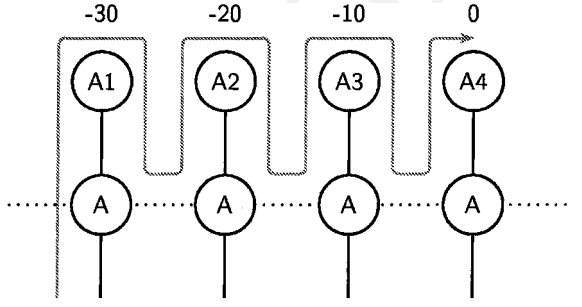


Fig. 3. Worst-case packet path with modified routing algorithm.

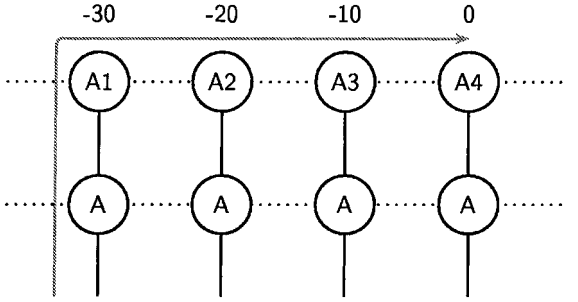


Fig. 4. Connecting arterial routers with a collector.

### 3.1.2. Oscillating and circulating packets

An apparent alternative to the topology restrictions described above is to allow both collector and arterial routers to forward packets in directions counter to what is permitted; for example, an arterial router forwarding a packet east or west rather than discarding it, or a collector or arterial router forwarding a packet out the port from which it was just received. These alternatives can cause network instability should a destination not exist:

1. A packet sent to an address between two routers (either collectors or arterials) will *oscillate* indefinitely.
2. A packet sent to an address that lies between two collectors and two arterials will *circulate* indefinitely.

Oscillating and circulating packets can be detected with a packet hop-count or time-to-live; the packet can be discarded when its hop-count limit is reached. However, since the Cartesian routing algorithm does not permit packets to retrace their routes and by discarding packets as soon as the destination is determined to be missing, the packet hop-count field and its associated processing is unnecessary.

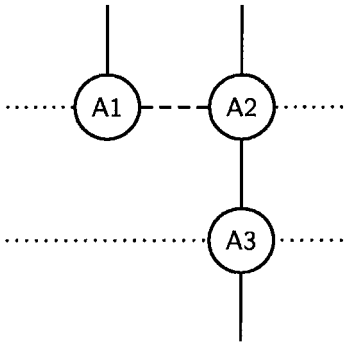


Fig. 5. Example of a virtual arterial (dashed line).

### 3.1.3. Virtual arterials

Although an arterial is not permitted to bypass a collector, there could be situations where it may be physically impossible for an arterial to span two collectors. In these situations it is necessary to allow arterial traffic to cross a collector to reach an arterial router for subsequent forwarding to the intended destinations. The segments of a collector employed in this manner are referred to as *virtual arterials*. In Fig. 5, the collector between A1 and A2 is used as an arterial for any southbound packets received by A1. However, any northbound packets received by A1 are routed over A1's physical arterial leading north.

## 3.2. Collector routers

A collector router is essentially the same as a linear router, with one major exception: packet destinations can be on a separate linear (collector) network.

### 3.2.1. Initial packet direction

The destination of a packet transmitted on a collector is either on the same collector or on a different one; the routing of the packet is determined by examinating the packet's destination address: a destination on the same collector has the same latitude as the transmitting router, whereas a destination on a different collector will have a different latitude. If the destination is on the same collector, the packet is transmitted according to the rules associated with the linear router.

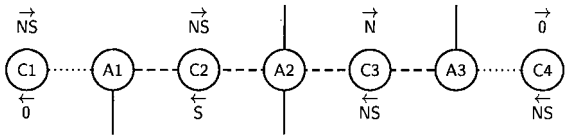


Fig. 6. Collector router ADIs (shown above and below each collector router).

However, if the destination is on a different collector, the collector router attempts to forward the packet in the direction of an arterial that leads to the destination. Forwarding to an arterial router requires that each collector router maintains two *arterial direction indicators* or ADIs: one showing the state of the eastern arterial, the other, the state of the western arterial (see Fig. 6). Collector routers not situated between arterials also have two ADIs, although only the one pointing in the direction of the arterial specifies the initial packet direction.

When deciding a packet's initial direction, the router first compares the packet's longitudinal destination address with its own address. The packet can be forwarded in the direction of the destination if the ADI indicates an arterial router in the same direction; otherwise, the packet must be forwarded in the opposite direction. (If both ADIs indicate that there are no arterials connected to the collector, the packet is discarded.) Since collector routers have no knowledge of the network's topology, the routing algorithm attempts to minimize the packet's path by forwarding the packet in the 'general direction' of the destination. It is assumed that in most cases, this approach is more likely to result in a shorter path than forwarding the packet in the opposite direction.

### 3.2.2. Packet routing

Packets can arrive on either port of a collector router. Packets intended for a different latitude are forwarded out the 'opposite' port from which they are received; the initial direction is determined by a packet's source router inspecting the ADI. Collector routers do not examine their ADIs when forwarding a packet to the arterial; this avoids infinite packet circulation between two adjacent collector routers with ADIs that point to a non-

existant arterial router situated between them. When the packet's destination latitude is the same as the router's latitude, the linear routing algorithm is employed to determine whether the packet should be kept, forwarded, or discarded.

### 3.3. Arterial routers

Inter-collector transmissions take place across an arterial. If a packet's destination is on a different collector, it must be sent across the source collector to an arterial router which directs the packet in the direction of the destination's collector.

Arterial routers are responsible for routing packets from one collector to another across the arterial. The routing algorithm for arterial routers is based upon the collector routing algorithm but is slightly more complex since packets can arrive from four possible directions (north, south, east, or west). To prevent packet oscillation, a packet can be forwarded in any direction other than the one from which it arrived. If the destination latitude is not equal to that of the arterial router, the packet must be forwarded north or south; otherwise it is forwarded east or west.

The arterial routing algorithm can be written as follows:

```
wait_for_pkt(&port, &packet);
if (packet.latitude == router
.latitude)
  if (packet.longitude == router
  .longitude)
    keep(&packet); /* Destination
    is this node */
  else
  switch(port)
  {
  case EAST:
    if (packet.longitude < router
    .longitude)
      forward(WEST, &packet);
    else
     discard(&packet);
    break;
  case WEST:
    if (packet.longitude > router
    .longitude)
      forward(EAST, &packet);
    else
      discard(&packet);
    break;
  default: /* From North or South —
  determine forwarding direction */
    forward(calculate_direction
    (packet.longitude), &packet);
  }
else
if (packet.latitude < router.lati-
tude)
  if (port != SOUTH)
    forward(SOUTH, &packet);
  else
    discard(&packet); /* From South
    : Destination is south of node */
else
  if (port != NORTH)
    forward(NORTH, &packet);
  else
    discard(&packet); /* From North
    : Destination is north of node */
```

If the packet's destination latitude is the same as that of the router and the packet is received on either arterial port (north or south), the packet must be forwarded east or west. The decision about which direction to forward the packet is determined in `calculate_direction()`, which employs the linear routing algorithm, returning `EAST` or `WEST`, depending upon the value of the packet's destination longitude and the longitude of the arterial router. Packets with destination latitudes that are outside the bounds of the network; that is, more northerly or more southerly than the limits of the network, are discarded by the boundary arterial routers.

#### 3.3.1. Reachability

In those situations where an arterial router does not have a physical connection across an arterial to another arterial router, it is necessary to employ virtual arterials. The virtual arterial allows an arterial router to send a packet across a collector to another arterial for forwarding towards the intended destination. Before a virtual arterial can be used, an arterial router must maintain a state that

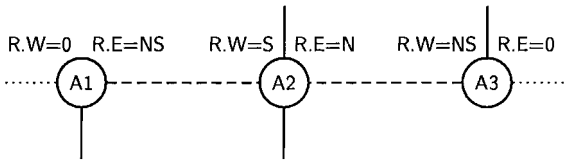


Fig. 7. Reachability: R.direction = 0|N|S|NS.

indicates whether an arterial can be reached traversing a collector and what directions are associated with the remote arterial (i.e., north, south, both, or neither). This state is referred to as *reachability*. Each arterial maintains the reachability of its east and west collectors, as shown in Fig. 7.

An arterial router's reachability is similar to a collector router's ADI, in that it gives the direction of an arterial. However, reachability differs in that it is used while a packet is traversing the network, as opposed to when a packet is about to be put onto the network.

### 3.4. Reducing path lengths

A Cartesian network topology is a grid of horizontal collectors and vertical arterials. All paths taken by packets between collector networks run horizontally (on a collector to an arterial), then vertically (across an arterial), and, finally, horizontally (on the destination collector). Although the vertical distance cannot be altered, diagonal connections between arterial routers can influence the horizontal distance travelled.

Since arterial routers have a single northbound and a single southbound connection, little is gained by connecting two arterial routers diagonally. However, by permitting arterial routers to support connections to more than one arterial router on an adjacent collector (i.e., north or south), it is possible to avoid some of the horizontal forwarding delays within a collector.

The approach that requires minimal state information is to limit the maximum number of arterial connections from an arterial router to six (three north and three south), where each arterial connection is either less-than, equal-to, or greater-than the arterial router's longitudinal address. The

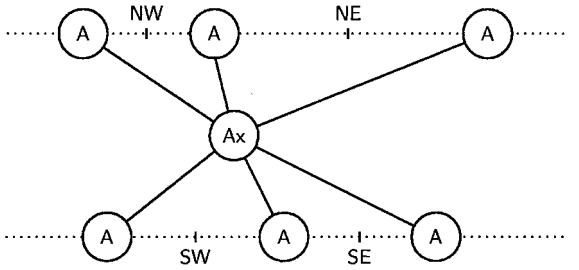


Fig. 8. Median longitudes for arterial router Ax.

arterial router can queue packets for north and south, as well as for north-west, north-east, south-west, and south-east.

If the network topology is such that there are few, if any, arterial routers due north or due south of each other, an alternate stateless design could have four connections per arterial router. In this implementation, the result of comparing the router's longitude with the packet's destination longitude is either less-than or greater-than-or-equal-to. A packet can be queued north-less-than, north-greater-than-or-equal-to, south-less-than, or south-greater-than-or-equal-to.

Any benefits associated with the above approaches can be lost when sending a packet to a destination that is slightly to the east or west of the arterial router closest to the destination. In these cases, the packet must overshoot, then be routed back, to the destination. This limitation can be overcome by modifying the six-arterial connection algorithm so that instead of comparing the destination longitude with the longitude of the arterial router, the destination longitude is compared with the eastern and western median longitudes on the adjacent collector. The median longitudes are the median points between the three arterial routers on the adjacent (i.e., north or south) collectors (see Fig. 8).

The values of the median points (both north and south) can be supplied to each router when the topology is created, or can be created dynamically during network initialization.

## 4. Network Initialization

The ADIs, virtual arterials, and reachability can be assigned statically when a network's topology is defined, or dynamically when the network is initialized. Dynamic initialization is more appropriate since it offers a more accurate picture of the current state of the network while lending itself to dynamic network reconfiguration and fault tolerance. Initially, all collector router ADIs and arterial router reachability state is undefined.

### 4.1. Collectors

Each collector router is responsible for updating both of its ADIs. During its initialization, a collector router polls its eastern and western neighbours with an *arterial-which-way* or AWW control message.

If the neighbour (another collector router or an arterial router) 'knows' that an arterial can be reached in the direction the AWW is taking, an *arterial-this-way* or ATW control message is returned; the neighbour only responds if it knows the current state of the arterial. The ATW control message indicates whether there is an arterial and if it leads to the north (ATW.N), south (ATW.S), both (ATW.NS), or neither (ATW.0). The ADI associated with the port from which the ATW has been received is updated with the value of the ATW. The ATW is then forwarded out the 'opposite' port.

### 4.2. Arterials

An arterial router can have physical connections on its arterials (north and south) and on its collectors (east and west); the state of the arterial router must be conveyed out its collectors and arterials. When an arterial router determines that it has at least one physical connection (north, south, or both), it issues an ATW control message out all of its arterials and collectors. The ATW control message carries the current state of the arterial router's arterial connections: one of north (ATW.N), south (ATW.S), both (ATW.NS), or neither (ATW.0); the most northerly and most southerly arterial routers issue ATW.S and ATW.N control messages out their southbound and northbound links, respectively.

An ATW can arrive at an arterial router from any arterial or either collector. When an ATW arrives from an arterial, the arterial router knows that it has a physical connection in that direction. An ATW from a collector indicates that there is a virtual arterial in the direction from which the packet has been received; the contents of the ATW indicate whether the virtual arterial leads to the north, south, both, or neither. The arterial router's reachability is updated to the value of the ATW that is received from the incoming collector. In order to determine the state of any virtual connections, each arterial router issues an AWW control message out its collector ports. The adjacent routers are expected to respond when their connection state is known.

An incoming ATW is always processed by the receiving arterial router. Whether the processing results in another ATW being sent to any or all of the 'other' ports depends upon the state of the router prior to its receiving the incoming ATW and the router's subsequent state. The state of the router is determined by its reachability and its physical connections.

For example, in Fig. 9, arterial router A has a physical connection to the south but has no reachability information. If an ATW.N arrives from the west, it means that the western collector has a virtual connection to the north. Since the arterial router has no physical connection to the north and (prior to the arrival of the ATW), could not reach the north via the collectors, the arterial routers to the south must be informed with an ATW.N that there is now a path to the north. Similarly, an ATW must be sent out the eastern collector; in this case, the indication is NS, since there is both a path to the north and the south.

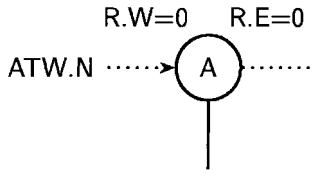


Fig. 9. State of router A prior to arrival of ATW.NS.

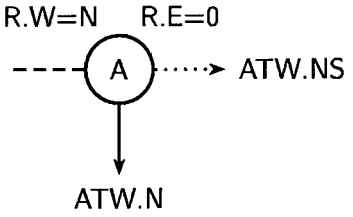


Fig. 10. State of router A after arrival of ATW.NS.

Finally, as shown in Fig. 10, the reachability to the west changes from 0 to N.

A similar cycle takes place for all the ATWs that are received by an arterial router. Packets can be routed off a collector (i.e., to the north or south), once an arterial router has a physical or virtual path to the north and south. An arterial router does not respond to AWW control messages until it has determined its reachability and the state of its arterial connections.

## 5. Fault tolerance

A Cartesian network with more than one arterial will have redundant physical and virtual paths between all arterial routers. The impact on the overall network of the failure of an arterial, a collector, or a router depends upon these redundancies.

### 5.1. Collector failure and recovery

Whenever a collector router detects a change of state in its connection, it must inform its neighbours. The failure of a collector or a collector router causes the adjacent collector router(s) to change their ADI for that port to '0' and to issue an ATW.0 out the opposite port (see Fig. 11). When the path between the two collector routers is re-established, they are responsible for forwarding the value of their ADI associated with the *opposite* port as an ATW. In both cases, each collector router that receives an ATW updates its ADI accordingly and forwards the ATW with its value unchanged.

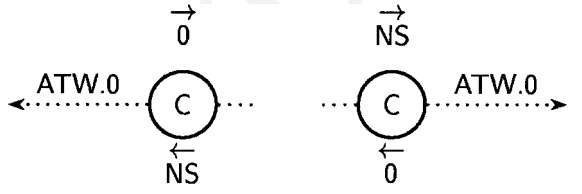


Fig. 11. Collector router actions after a collector failure is detected.

### 5.2. Arterial failure and recovery

The state of an arterial router depends upon its reachability and its physical connections. The router's state can change when it receives an ATW or detects a physical failure. Should the router's state change, it may be required to send an ATW out its arterials, its collectors, or both.

In Fig. 7, if A1's southbound arterial fails, it must forward an ATW.0 out its eastern port (i.e., the collector leading to A2); A1's reachability does not change (meaning that the ADIs of the collector routers to the west of A1 need not be updated). All collector routers that receive the ATW.0 must update their ADIs from 'S' to '0', since A1 no longer has a path to the south.

When the ATW.0 reaches A2, A2 must change its western reachability from R.W = S to R.W = 0. There is no need to inform the collector routers on the collector between A2 and A3, since their western ADI's and the reachability of A3 refer to 'NS', the paths support by A2.

Should A1's southbound arterial recover, A1 must send an ATW.S out its eastern port. As with the previous transmission of the ATW.0, the collector routers must update their ADI's and A2 must change its reachability from R.W = 0 to R.W = S.

### 5.3. Islands

A collector is a potential single point of failure. When a collector fails, all communications for routers on the same latitude but beyond the failure cease, since packets for destinations on a collector cannot be routed off the collector. In those situations where a collector is reduced to a single arterial router and the arterial router has no physical

north or south connections, the collector is said to be an *island*.

## 6. IPv6 geographic addresses

A barrier to the adoption of Cartesian routing is the availability of an addressing structure. Many limited areas, such as offices, campuses, or even metropolitan areas often exhibit a natural ordering that fit naturally into an ordering that lends itself to Cartesian routing. On a larger scale, routers and other devices can be identified by latitude and longitude. The availability of low-cost GPS receivers would allow these routers to 'learn' their location.

Latitude and longitude are commonly expressed in terms of degrees, minutes (60 min in a degree), and seconds (60 s in a minute); with longitude spanning 180°W (−180) to 180°E (+180) through 0°, and latitude 90°S (−90) to 90°N (+90). When longitude and latitude are expressed in terms of seconds, they require 21 and 20 bits of storage, respectively. The existing 32-bit Internet address structure is insufficient to represent both longitude and latitude with accuracies of more than 5 and 10 s, respectively (ignoring the storage required for the packet class). A geographic address structure for the 32-bit IP address structure is described in [9]; however, it is limited to the continental United States. In [4], a design for storing geographic addresses in the DNS is proposed using 256 byte strings.

### 6.1. IPv6 address structure

The IPv6 address structure is 128 bits in length. As with IPv4, some of the bits in the address structure are prefix-bits, equivalent to the packet class, used to identify the address type. IPv6 supports from 3 to 10 prefix-bits; for example, `010`, is reserved for aggregatable global unicast addresses, `11111111`, for multicast addresses, and `000010`, for IPX allocation. Geographic addresses have the 3-bit prefix `100`.

With only a 3-bit prefix, geographic addresses have 125 bits remaining to represent an object's location. Although the organization and subsequent interpretation of these bits has yet to be finalized, three possible structures are presented and considered.

#### 6.1.1. Complete geographic

At one extreme, the entire 125 bits could be devoted to representing an object's location. Since latitude refers to the complete globe and longitude a hemisphere, a location could be divided into a 63-bit longitude and a 62-bit latitude. Since the equatorial circumference of the Earth is 40,075 km, and the polar circumference is 39940 km, a single geographic address would identify an area approximately $4 \times 10^{-10}$ cm by $4 \times 10^{-10}$ cm at the equator.

As this is probably more accurate than can be justified, devoting all 125 bits to identifying an object's location is excessive, suggesting that other structures be considered.

#### 6.1.2. Aggregatable address

The IPv6 aggregatable global unicast addresses are divided into two 64-bit structures: three 'aggregator levels' (totalling 61 bits), and an interface id (64 bits). The interface id uniquely identifies an interface that is connected to a link. If the geographic address were to support a 64-bit interface identifier, there would be 61 bits remaining to represent the latitude and longitude of an object.

Given the relationship between latitude and longitude, the 61 bits available could be divided into a 31-bit longitude and 30-bit latitude. At the equator, a 31-bit longitude and 30-bit latitude would identify an area approximately 2 cm by 1.8 cm.

#### 6.1.3. Interface identifier

A third approach is to combine the IPv6 aggregatable global unicast address with a geographic address as the interface identifier.

In the proposed 64-bit interface identifier structure, most identifiers are expected to conform to IEEE EUI-64 [8]. Other address structures, such as the IEEE 802 48-bit address, can be supported; in the case of the 48-bit MAC address, the address is divided in half, with the 16-bit hexadecimal value `FFFE` separating the two halves.

In keeping with the representation of 48-bit MAC addresses in EUI-48, an object's location could be represented by 48 bits; for example, 24-bits for latitude and 24-bits for longitude. If these values are used, the equatorial area identified by the addresses would be some $2.4 \times 10^2$ cm by $1.2 \times 10^2$ cm.

## 7. Summary

A limitation of existing hierarchical routing schemes is the size and maintenance of the routing tables. A typical layer 3 hierarchical router requires between O($n$) (for an unsorted routing table) and O(log($n$)) (for a sorted routing table) to find a network address in its routing table, although this can decrease using caching and other techniques. This paper has proposed an alternative to hierarchical routing, known as Cartesian routing; routing packets based upon the packet's destination location, relative to that of each router. In addition to eliminating the storage requirements for routing tables, a Cartesian router requires O(1) time to determine how it should dispose of an incoming packet.

The routing algorithm relies upon a structured topology with routers assuming specific roles based upon their position within the network. By restricting the topology of the network to horizontal (collector) and vertical (arterial) links, minimal state is required. Packets are kept, forwarded, or discarded as each router compares its address with the packet's destination address; as a result, the algorithm eliminates the need for routing tables.

Allowing an arterial node to have multiple arterial links both north and south, and by using collectors as virtual arterials, a degree of fault tolerance is achieved. As long as a path exists for a packet between two routers, communications can be maintained. When an arterial router's physical connections to the north, or south, or both, fail, it can continue as an arterial router using its horizontal virtual arterials, as specified by the router's reachability. A link failure within a collector or the failure of a collector router is fault intolerant should a packet's path cross the failed link or router.

A limited number of control messages is required when initializing the network and to handle network faults. To allow packet transmission to destinations on different collectors, all collector routers maintain a pair of ADIs. Each ADI is associated with one of the router's ports (east or west) and indicates whether an arterial can be reached via that port. The value of each ADI is determined when the router receives an ATW control message from one of its ports. As part of a collector router's initialization sequence, it issues an AWW control message; the response to which is an ATW. Arterial routers are responsible for informing their neighbours with ATWs should a state change occur because of a change in its connectivity.

Prior to the development of IPv6 and its 128-bit address structure, identifying an object by its location was not easily achievable using IP, since the 32-bit address offered too coarse a resolution. However, with 128 bits, geographic locations can easily be represented by encoding a location as a latitude and longitude pair; three possible encodings were considered, giving equatorial accuracies of about $10^{-10}$, 1 and $10^2$ cm$^2$.

The principal limitation associated with the routing algorithm is its requirement for a structured topology. New stations can be added in an ad hoc fashion; however, they must join an existing network and be placed in a unique location with their address being specific to the location in question. Other networks exhibit rules regarding the addition of new stations: the Ethernet requires a minimum station separation and all networks require stations to associate with unique unicast addresses.

In addition to the issue of the required topology, the possible penetration or acceptance of Cartesian routing will depend upon a number of factors, including:

- *Changes in the concept of what constitutes a 'provider'*. At present, providers are not associated with geographic regions, and in some cases, several providers can share a common region. As a result, routing within a region may be restricted to an area that is serviced by a single provider.

Alternatively, organizations within a region may opt to select a single geographic provider.

- *Adaptation of existing networks to handle Cartesian routing*. The existing Internet exhibits no topological structure that could be considered as suitable for geographic routing. If existing networks are to support geographic routing, it will be necessary either to rewire them or to employ techniques such as tunnelling.

At present, the authors are implementing a visual representation of the Cartesian routing algorithms. Possible hardware designs of the collector and arterial router algorithms are being explored with the objective being a limited area demonstration. Two techniques for linking Cartesian networks in a larger geographic area are currently under development and will be described in subsequent papers.

## Acknowledgements

The research described in this paper is supported by an operating grant from the Natural Sciences and Engineering Research Council of Canada.

## References

[1] D. Comer, Internetworking with TCP/IP, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[2] D. Comer, Computer Networks and Internets, Prentice-Hall, Englewood Cliffs, NJ, 1996.
[3] S. Deering, R. Hinden, Internet Protocol, Version 6(IPv6) Specification, RFC 1883.
[4] C. Farrell, M. Schulze, S. Pleitner, D. Baldoni, DNS Encoding of Geographical Location RFC 1712.
[5] L. Hughes, Teaching Data Communications to Computer Science Students, SIGCSE Bulletin 21 (1) (1989).
[6] L. Hughes, Introduction to Data Communications: A Practical Approach, Jones and Bartlett Publishers, Boston, 1996.
[7] C. Huitema, IPv6 – The New Internet Protocol, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1997.
[8] IEEE, Guideline For 64-bit Global Identifier (EUI-64) Registration Authority. see http://stabdards.ieee.org/db/oui/tutorials/EUI64.html.
[9] T. Imielinski, J.C. Navas, GPS-based geographic addressing, routing, and resource discovery, Comm. ACM 42 (4) (1999) (see also, GPS-Based Addressing and Routing, RFC 2009).
[10] M. Reid, Personal communication, March 2000.
[11] C. Semeria, Understanding IP addressing: everything you ever wanted to know, April 1996. www.3com.com/nsc150/302.html.

**Dr. Larry Hughes** is an Associate Professor in the Department of Electrical and Computer Engineering at Dalhousie University. His research interests include computer communications, embedded systems, and renewable energy. He is the author of over 50 papers and several textbooks. Dr. Hughes obtained his Ph.D. in Computer Science from the Computing Laboratory at the University of Newcastle upon Tyne in 1988.

**Mr. Omid Banyasad** received a B.Sc. in Computer Hardware Engineering from University of Tehran in 1993. He obtained an M.Sc. in Computer Science from DalTech, Dalhousie University in January 2000. His research interests include computer networks, the next generation of The Internet, visual programming languages for robot control and visualization systems. He is currently working towards his Ph.D. at Dalhousie University.

**Mr. Evan Hughes** is currently an undergraduate computer science student at Carleton University in Ottawa. He plans to graduate in August 2000.