Internet Routing

Thomas Narten
Department of Computer Sciences
Purdue University
West Lafayette, Indiana 47907
narten@cs.purdue.edu

Abstract

Comprising an estimated 60,000 hosts, the DARPA Internet is the largest existing internet. This paper traces the routing information protocols used by Internet gateways to build routing tables that define the paths datagrams traverse as they travel between end systems. We articulate the weaknesses and limitations of the most commonly used routing protocols, including RIP, GGP, and HELLO and examine how the protocols interact with each other and with EGP. Finally, we trace the evolution of routing as the Internet has grown from a single backbone (ARPANET) to its present inclusion of the ARPANET, Milnet, and NSFnet cross-country networks.

1 Introduction

Networking has revolutionized computing. Scientists use networks to exchange data, disseminate research results, and collaborate with others. Because no one technology satisfies the diverse needs of users, thousands of independent networks are scattered around the world. Through the technology known as *internetworking* [Pos80], multiple independent networks can be joined into a single virtual network called an *internet*. Internet technology abstracts away the details of the underlying physical connections and provides users with the illusion of connecting to a single, homogeneous network.

In the largest internet, the connected TCP/IP Internet [HHS83,JLF*86], an estimated 60,000 hosts communicate using the TCP/IP protocol suite [Com88]. As the Internet has grown, so have the mechanisms it employs in selecting the paths that carry datagrams from their source to their destination. The process of path selection is known as *routing*. In the early 1980s, the size and topology of the Internet was sufficiently comprehensible that route tables could be managed by such ad hoc, manual mechanisms as configuration tables. As the Internet grew, however, researchers invented protocols that automated the propagation of routing information and allowed new sites to join the Internet and communicate instantly with other sites.

This paper focuses on the mechanisms used to route IP datagrams through the Internet[1], articulating the protocols, interactions among protocols, and the scattered pieces of information needed to understand how the Internet routes datagrams between end systems. Section 2 reviews the Internet routing architecture and reviews the strengths and weaknesses of vector-distance and link-status based routing protocols. Section 3 reviews autonomous systems, the Exterior Gateway Protocol (EGP), and the Core system. Section 4 presents common routing protocols used within an autonomous system, including RIP, GGP, HELLO. In Section 5, we chronicle the evolution of routing within the Internet, covering the ARPANET, MILNET, and the first two phases of NSFnet. Section 7 presents conclusions, and the Appendix documents several events that lead to Internet-wide routing failures.

2 Background

The Internet architecture clearly distinguishes *hosts* from *gateways*. Hosts are computers that execute application programs on behalf of users. They may be time-sharing systems, batch systems, workstations, or personal computers. Although hosts may depend on network services, their primary function is user service. In contrast, gateways are the building blocks that connect networks into internets. A gateway connects to two or more networks. It receives datagrams from hosts and

[1]The term *internet* refers to any internet; we use *Internet* to refer specifically to the connected TCP/IP Internet.

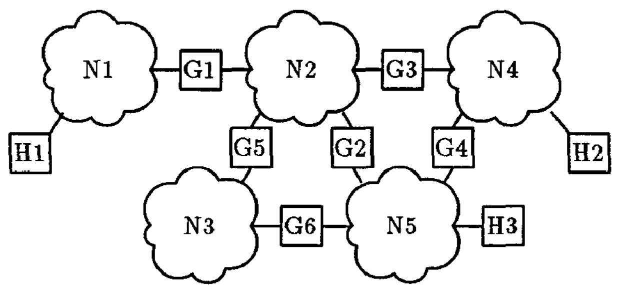

Figure 1: Sample internet with hosts, networks, and gateways.

gateways on one network and forwards them to hosts or gateways on another network. The proper functioning of the internet depends on the correct operation of gateway hardware and software.

Gateways process datagrams in a store-and-forward manner. When a datagram D arrives at a gateway G, G selects another gateway, N, that is (1) closer to the datagram's destination and (2) directly reachable from G. G queues the datagram for transmission to N using the appropriate network interface hardware. N is called the *next hop for datagram D* and the process of selecting a next-hop host or gateway is known as *routing.* Figure 1 illustrates routing. The example internet consists of five networks interconnected by six gateways (Gi, $1 \leq i \leq 6$) and three hosts (Hi, $1 \leq i \leq 3$). Suppose that H1 wants to send a datagram to H2. Because it cannot reach H2 directly, H1 sends the datagram to gateway G1. Although G2, G3, and G5 are candidate next hops, G1 forwards the datagram to G3 because of its proximity to H2. G1 sends the datagram directly to G3 across network N2, and G3 sends the datagram directly to H2 across network N4.

To adjust dynamically to changes in topology, gateways run *routing information protocols* that disseminate topological information about the internet. Through routing protocols, gateways learn which networks are currently reachable, and the appropriate next-hop gateway to use in reaching a given destination.

2.1 Host Routing

Although hosts send datagrams to gateways, they should not participate in the same routing information protocols used among gateways. First, the number of hosts on a network typically exceeds the number of gateways by an order of magnitude or more, and the exchange of information needed by routing protocols consumes resources. Second, routing protocols continue to evolve, and host software must be updated to take advantage of each new routing algorithm. Changing software on a few gateways is easier than changing software on hundreds of hosts. Finally, some host computers, such as single-tasking personal computers, lack facilities needed to participate in gateway routing protocols. Thus, mechanisms that propagate routing information from gateways to hosts should be independent from those used to propagate such information among gateways.

To route datagrams, a host need only maintain a set of pointers to neighboring gateways and a cache of recently used routes. When sending a datagram, a host searches its cache for a next-hop gateway to use in reaching that datagram's destination. If the cache lookup fails, the host creates a new entry, filling in the gateway field with one of the "default" gateways in its set. In Figure 1, a single gateway separates H1 from the rest of the world, and the need for a cache of routes is not obvious. Host H2, however, shares a network with two gateways. If it sends a datagram destined for H3 to gateway G3, G3 might route it to G4, which H2 can reach directly.

The Internet architecture includes a redirect mechanism that gateways use to notify hosts when they have chosen the "wrong" gateway. In our example, gateway G3 sends H2 an ICMP redirect [Pos81] telling it to forward traffic destined for H3 through G4. Upon receipt of a redirect, a host updates the gateway field of the corresponding cache entry for that destination.

The redirect mechanism can only be used between gateways and hosts, because it relies on a gateway's ability to determine which neighboring host or gateway forwarded the datagram to it. In particular, when sent by a host, the network portion of the datagram's source address will match that of one of the gateway's connected networks. Whenever a gateway forwards a datagram to a next-hop gateway on the same network as datagram's source, it knows that the host can reach that gateway directly and sends the host a redirect.

2.2 Gateway Routing Protocols

Gateways run routing protocols to learn the topology of the internet. Roughly speaking, existing routing protocols fall into two general classes. "Link-status protocols" propagate the status of the actual topology of the internet. In contrast, gateways running "vector-distance protocols" exchange routing tables with one another. The following subsections review the two classes of algorithms.

2.3 Vector-Distance Protocols

Vector-distance algorithms, also known as Bellman-Ford algorithms [BG87], were first used in the original ARPANET [MFR78]. Update messages consist of (destination, metric) pairs. Conceptually, a gateway generating update messages advertises routes for the destinations it can reach, with a metric indicating the cost of the path to the destination. From the received updates, a gateway selects for each destination the next-hop gateway advertising that destination at the lowest metric. A special metric value of *infinity* denotes paths that are no longer reachable and should not be used.

As an example, suppose that the gateways in Figure 1 run a vector-distance protocol whose update metric corresponds to the number of gateways traversed in reaching the destination[2]. Gateways G3 and G4 advertise network N4 at metric zero. G2 and G6 advertise N4 at metric one, and G5 advertises the network at metric one. Gateway G1, receiving updates for N4 from G3, G2, and G5 at metrics of zero, one, and one respectively, selects G3 as its next hop for N4.

Details of individual vector-distance protocols depend on their specific implementations. In some implementations the sender returns updates only in response to requests; in others, gateways send periodic, unsolicited updates. Likewise, some implementations assume fixed link costs, while others vary a link's cost to reflect load or delay. Finally, in some implementations, the sender measures the cost of the link and adds it to the routing metrics prior to sending an update, while in other implementations, such functions are delegated to the receiver.

The relative simplicity of vector-distance algorithms contributes to their wide popularity. They use only local information, and gateways only exchange information with neighbor gateways. However, vector-distance algorithms suffer from what is known as the "counting-to-infinity" problem. If a link fails, or its cost increases suddenly, routing loops can develop because some gateways still send updates containing old information. Moreover, loops can persist for a significant amount of time.

For example, consider the four gateways in Figure 2. Initially, gateway G1 advertises a route for N1 at zero hops, and G2 advertises the route for N1 to G3 at one hop. G3, in turn, advertises the route at metric two. If G1 stops sending updates, G2 will conclude that N1 has become unreachable through G1 and adopt the path advertised by G3 at metric two, creating a "ping-pong" routing loop between neighbors G2 and G3. During the next update exchange, G3 will advertise the route at metric three, and G2 continues using the route. Each gateway's metric increases with subsequent updates, and the loop persists until the metric counts up to infinity.

Ping-pong loops form between adjacent gateways because neither is aware that they are both routing datagrams for a common destination through each other. Loops between adjacent gateways are one instance of a more general problem with vector-distance protocols. Specifically, loops form when gateway G receives an update and switches the route for a given destination to a new next-hop gateway unaware that the path that the next-hop gateway uses to reach that destination actually travels through G. Stated more formally, suppose that gateway G1 has a loop-free path to destination DEST. In order for loops involving G1 and DEST to form, the following conditions must be met:

[2]This metric is known as a hop count.

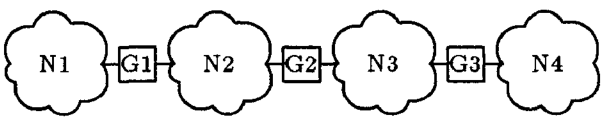

Figure 2: A ping-pong loop for destinations on network N4 forms between gateways G1 and G2 after gateway G3 crashes.

Condition 1 *Gateway G1 receives an update from one of its neighbors G2 reporting a current metric for destination DEST that is less than G1's current metric to DEST.*

Condition 2 *G1 is included in G2's path to DEST.*

Loop formation can be prevented by taking appropriate action when the two conditions are met. For instance, if a gateway sending a routing update to a neighbor identifies those destinations whose paths include the receiving gateway, the receiving gateway can ignore information about such routes, preventing condition 2. The technique popularly known as *poisoned updates* does just that: when a gateway sends an update to one of its neighbors, all routes through that neighbor are given a metric of infinity, forcing the receiving gateway to ignore the information. Likewise, in the related technique called *split horizon*, gateways omit from routing updates entries for those destinations it routes through the gateway targeted by the update. With vector-distance protocols, however, the sending gateway can only identify the first hop of a path to a destination and the two techniques can only prevent ping-pong loops; loops involving more than two gateways can form. The problem stems from updates that are based on old information. The following definition clarifies the problem.

Definition 1 *A routing update U1 is based on update U2 with respect to destination DEST if the information in U1 regarding DEST was derived from information contained in U2.*

Intuitively, loops can form when a gateway processes an update based on an update it sent previously. For example, ping-pong loops form when a gateway G receives an update from its neighbor that was based on a previous update sent by G to that neighbor. With vector-distance protocols, news travels approximately one hop per update interval. Thus, in large networks, a gateway might receive updates based on an update it generated several update intervals ago, leading to the following proposition:

Proposition 3 *If a pair of gateways generate update messages U1 and U2 at times T_{U1} and T_{U2} respectively, than there is some length of time T such that: if $T_{U1} - T_{U2} \geq T$, then U1 is not based on U2.*

Intuitively, the proposition states that information in an update remains in the network for a finite time less than some value T. *Hold-down* [MRR78] is a heuristic that takes advantage of the proposition. Whenever a gateway's metric to a destination increases, it ignores further information about that destination for a time period of length T. Hold-down guarantees that any routing updates meeting Condition's 1 and 2 will be ignored.

Although hold-down timers can prevent the formation of certain types of loops, they suffer from two deficiencies. First, the hold-down time T is proportional to the update propagation time across the network and may be several minutes long. Second, networks using hold-down are slow to adapt to topology changes, a self-defeating property for an adaptive routing algorithm. If the metric of a path currently in use increases or reaches infinity, for instance, a gateway continues using it even if a better path is available. Using hold-down to prevent all loops, however, might be too ambitious. Indeed, a short value for the hold-down timer can prevent ping-ping loops, while longer timers prevents loops of increasing path length. Hold-down timers and their problems are described in detail in [MRR78].

2.4 Link Status Protocols

Link-status protocols propagate the status of the actual physical topology of the internet to each gateway. Update messages consist of (link, metric) pairs, where the link identifies a pair of adjacent gateways, and the metric gives the cost of using that link. A metric's value might be fixed, or give a dynamic estimate of its load, transmission delay, etc. Periodically, a gateway controlling the link broadcasts update messages to all participating internet gateways. From the database of link costs, each gateway constructs a tree of shortest paths leading to each destination.

Because update messages are broadcast to each gateway, each gateway builds its routing tables from the same information, reducing the probability that routing loops will form. Thus, link-status protocols are less vulnerable to loop formation than vector-distance algorithms. In practice, link-status protocols are not entirely loop free. In a large internet, for instance, broadcast update messages may propagate slowly, and various gateways might receive an update at different times. Thus two gateways might compute routing tables based on a different set of updates. In addition, the cost of sending updates is significant and does not scale well to internets comprised of thousands of networks and gateways. To insure that each gateway builds its tables from the same information, update messages must be carried in reliable broadcasts guaranteed to reach every gateway, and consecutive updates from the same gateway must be processed serially. Link-status algorithms were first used within the ARPANET [MRR80].

3 EGP and the Core

When the Internet formed, Bolt, Beranek, and Newman Inc. (BBN) administered and operated the ARPANET backbone network. In an internet, however, no single organization could manage all the constituent networks and gateways, and Internet architects created *autonomous systems*, administrative regions within which sites manage routing themselves. Within an autonomous system, sites use a private interior gateway protocol (IGP) to propagate information about routes. To learn about networks in other autonomous systems, gateways between autonomous systems run the Exterior Gateway Protocol (EGP) [Mil84]. Gateways use EGP to find out how to reach networks located in other autonomous systems, and to advertise the existence and reachability of networks located within their autonomous systems. IGPs can be used only within a single autonomous system and EGP is the sole means by which route information from one autonomous systems may flow into another.

Three aspects of the EGP reachability protocol should be noted. First, EGP isolates sites from one another. Gateways running the EGP protocol negotiate the values of parameters specifying how frequently reachability information can be exchanged, allowing a gateway to control the frequency at which it will be interrupted with updates from gateways in other autonomous systems. Moreover, a site's internally managed routes do not depend on EGP or on sites located in other autonomous systems.

Second, EGP is a vector-distance algorithm; gateways associate a metric with each route. Unlike other vector-distance algorithms, however, EGP update messages allow a sending gateway to specify a third gateway through which a destination is reachable. Intuitively, conventional distance-vector algorithms generate update messages of the form "I can reach network X at metric K", whereas EGP updates carry information of the form "network X is reachable through gateway Y at metric K".

Finally, EGP is a reachability protocol rather than a routing protocol because the EGP specification leaves interpretation of the metric undefined. With the exception of an infinity metric, all metrics indicate that a destination is reachable, and the comparison of metrics in updates generated by gateways in distinct autonomous systems is explicitly undefined. The absence of a universal metric interpretation is a fundamental aspect of the Internet routing architecture.

In practice, the metric definition restricts the Internet topology to being a tree. In a tree, only one path exists between any two autonomous systems, and a gateway is never forced to compare metrics advertised by gateways in different autonomous systems. When such a comparison is necessary, the metric definition specifies that either gateway should work.

Because the ARPANET was the major cross-country network in the Internet, it was natural to place it and the other BBN-operated networks into a single autonomous system. The BBN-operated autonomous system was dubbed the *Core System*. Gateways in the Core System maintain complete routing knowledge about all IP networks. Core gateways may not use default routes, and for any given destination must either forward the datagram, or assert authoritatively that the destination is unreachable. A second important characteristic of the Core concerns propagation of EGP-derived information. Core gateways are the only Internet gateways authorized to advertise routes reachable through other autonomous systems. In contrast, EGP's so-called "third party rule" prohibits non-Core gateways from advertising routes for networks not belonging to their autonomous systems. The third-party rule effectively restricts the topology of the Internet to a tree in which all autonomous systems connect directly to the Core.

The following section reviews four common IGPs used within autonomous systems.

4 Interior Gateway Protocols

4.1 Gateway-to-Gateway Protocol

The first Core gateways [HS82] ran on DEC LSI-11 hardware and connected to BBN-operated networks such as the ARPANET [MW77] and SATNET [JBH78]. In addition, some Core gateways also connected to local networks at the sites where they were housed. LSI-11 gateways execute the Gateway-to-Gateway Protocol (GGP) [HS82] to exchange information about the reachability of IP networks. The following points highlight GGP:

- GGP is a vector-distance protocol. GGP metrics specify the number of gateway hops on the path to the destination, and the infinity value of 255 denotes destinations that are unreachable.

- GGP uses echo request and reply messages to monitor neighbor reachability. Gateways use a K out of N algorithm to declare neighbor gateways UP or DOWN. In the earlier Core gateways, gateways generated echo messages every 15 seconds, and gateways in the DOWN state were declared UP upon receipt of 2 responses out of the preceding 4 echo requests; gateways in the UP state were declared DOWN if 3 of preceding 4 messages remain unanswered.

- Instead of sending periodic updates, gateways generate GGP update messages in response to the following events:
  - A change in the status of a directly connected network.
  - A change in the reachability status of a neighboring gateway.
  - A change in its routing tables resulting from a received GGP message.

  A restarting gateway can request updates from neighboring gateways.

- GGP does not age routes. A route previously reachable through a next-hop gateway becomes unreachable when the next-hop gateway sends an update message that omits the route.

- Because GGP does not age routes, gateways use sequence numbers and retransmissions to exchange routing updates reliably.

- Gateways send GGP messages as unicast datagrams. Core gateways do not send multicast or broadcast routing messages.

- On startup, a gateway consults a configuration database to obtain a list of peers, gateways with which it exchanges GGP messages. In addition, a gateway adds new neighbors to its list of peers when it receives GGP messages from gateways not already on its list.

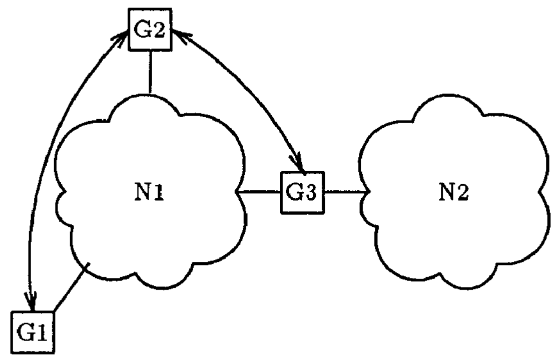

Figure 3: The GGP extra-hop problem arises when gateways G1 and G3 do not run GGP with each other. Datagrams destined for hosts on network N2 travel along the path G1–G2–G3.

The nature of GGP updates can lead to the phenomenon known as the *extra-hop problem*. Consider the configuration in Figure 3, where three gateways connect to common network N1. Gateway G2 runs GGP with both G1 and G3, but G1 and G3 are non-peering gateways. If G3 advertises a path for network N2 to G2, and G2 advertises the path to G1, the update messages sent by G2 to G1 contain only enough information for G1 to conclude that G2 has a path to N2, and the packets it sends to hosts on network N2 travel along the path G1–G2–G3 rather than from G1 to G3 directly. The piece of information missing from GGP updates is that G2's best path is through G3, which G1 can reach directly.

The extra-hop problem can be eliminated if every gateway on a network runs GGP with every other gateway on the same network. However, because the ARPANET does not support the broadcasting or multicasting of IP datagrams, such a solution does not scale well with an increasing number of gateways. By late 1988, the Core system included approximately 45 LSI-11 gateways and those gateways ran GGP with all other Core gateways on a common network.

4.2 Routing Information Protocol

As part of the Berkeley Standard Distribution (BSD) of UNIX[3] [QSP85], researchers at the University of California at Berkeley developed the Routing Information Protocol (RIP). The basic outline of the protocol was derived from the Gateway Information Protocol used in the PUP protocol suite [BSTM80]. Berkeley distributed RIP in the UNIX daemon *routed*, and the (undocumented) protocol became a de facto standard. Only recently has the protocol been formally documented [Hed88]. The following points highlight the protocol:

[3]UNIX is a Trademark of AT&T Bell Laboratories

- RIP is a vector-distance algorithm. RIP metrics denote hop counts and the infinity metric of 16 indicates unreachable destinations.
- RIP sends unsolicited update messages every thirty seconds. When supported, RIP uses the broadcast capability of the underlying physical network.
- RIP ages routes; gateways delete those routes not appearing in any received update over a six-update (180 second) time interval.
- RIP messages allow booting gateways to request updates from neighboring gateways. In normal operation, gateways listen for periodic update messages.
- To propagate changes quickly, RIP gateways generate "triggered" or "flash" updates when they receive an update containing information that changes their routing tables. Flash updates can spawn a flurry of routing updates, especially when metrics rise because of the counting-to-infinity phenomenon.

RIP was designed for use in a LAN environment. To limit the effects of the "counting to infinity" problem, its metric was kept deliberately small. Moreover, although routing loops can form, the combination of flash updates, highly reliable delivery, and high bandwidth associated with LANs reduces the duration and resource cost of such loops. In addition, in a LAN environment, the link cost between any two machines is essentially constant, justifying hop counts as a metric. Its authors make no claims as to its suitability for use in non-LAN environments.

Finally, UNIX systems define the all-zeroes destination to be a *wildcard*[4] route, a path for datagrams for which no explicit route exists. Wildcard routes are particularly useful within an autonomous system, where updates might contain explicit routes for local networks and a wildcard route leading to a gateway running EGP with the Core.

4.3 HELLO

The HELLO protocol [Mil83] was developed as part of the PDP-11 based Fuzzball gateway [MB87], and Fuzzballs were used in the first NSFnet backbone. The following points highlight the protocol:

- HELLO is a vector-distance protocol and metrics denote the time delay datagrams encounter as they travel along the path to a destination. An infinity

[4]In UNIX terminology, wildcard routes are called default routes.

metric of thirty seconds denotes unreachable destinations.

- The protocol can be used over point-to-point or broadcast networks.
- Gateways generate periodic, unsolicited update messages. The delay between successive updates depends on the link capacity and ranges from a minimum of eight to a maximum of thirty seconds.
- Gateways use a hold-down timer of two minutes.

4.4 Butterfly Gateways

As a replacement for the LSI-11 Core gateways, BBN developed a multiprocessor-based gateway called the Butterfly. Butterfly gateways use a link-status IGP to propagate routing information about networks that connect directly to Butterfly gateways and send update messages to all other Butterfly gateways in a multicast-like fashion. Update messages consist of two parts: a list of attached network interfaces in an UP state, and a list of neighboring gateways together with the cost of reaching that neighbor. Although link costs are fixed, each link can be assigned a separate cost. Typically, network administrators assign low costs to high-bandwidth, low-delay links such as Ethernets, and higher costs to paths crossing ARPANET or satellite links.

5 Internet Routing

Internet routing has evolved through roughly five time periods. The first period, in which the ARPANET was the sole backbone Internet network, covered the late 1970s through 1983. The second period began in October, 1983, when the ARPANET was split into two networks, the ARPANET and MILNET. The third period began in 1986, when the first phase of the NSFnet project became operational. In NSFnet's first phase, 56 kbps links connected "Fuzzball" gateways at six supercomputer sites across the country. The fourth period began in June 1988, when Merit assumed operation of the NSFnet backbone. Finally, the LSI-11 mailbridges separating the ARPANET and MILNET were replaced with Butterfly gateways in December, 1988.

5.1 ARPANET Core

In the early 1980s, the ARPANET was the Core System's backbone network. Core gateways exchange routing information via GGP, and they maintain routes to every other network in the Internet. Figure 4 illustrates how routing information propagates from one

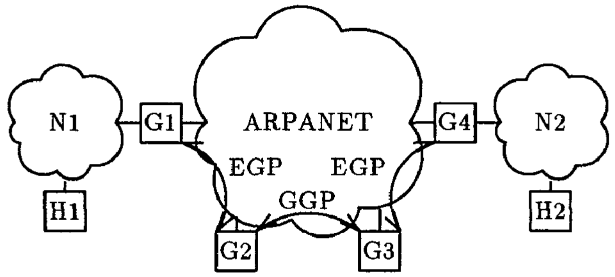

Figure 4: Gateway G1 uses EGP to advertise a route for network N1 to Core gateway G2. G2 uses GGP to propagate the route to G3. Finally, G3 uses EGP to advertise a route for N1 to G4.

autonomous system to another. Non-Core gateways G1 and G4 connect to the ARPANET and to a LAN in their respective autonomous systems. Gateway G1 runs EGP with G2, informing G2 that network N1 is reachable through G1. Within the Core, GGP propagates information about N1 to other Core gateways, including G3. Finally, through EGP, Core gateway G3 informs gateway G4 of N1's reachability and learns of N2's reachability. Routing information about network N2 propagates back to G1 in an analogous fashion.

The above example illustrates the mechanism by which gateways obtain routes to all other Internet sites. Unfortunately, the path datagrams travel in reaching those destinations can be suboptimal. In our example, datagrams traveling from host H1 to H2 follow path G1-G3-G4 rather than traveling from G1 to G4 directly; likewise, datagrams traveling from H2 to H1 travel along path G4-G2-G1. The problem lies with GGP, which is only able to convey information of the form: "I can reach network X". Thus, when G2 advertises routes to G3 via GGP, G3 is unaware that it can reach G2's next-hop gateways directly. In our example, both G2 and G3 are on the ARPANET, and both can send traffic directly to G1. This phenomenon is commonly referred to as the GGP extra-hop problem.

To eliminate the extra-hop problem, BBN designated three Core gateways to be EGP servers, and sites were directed to peer with the EGP servers, rather than arbitrary Core gateways. Moreover, each site was required to simultaneously peer with more than half of the EGP servers, assuring that any two ARPANET sites would peer with at least one common EGP server. Because the common server directly exchanges EGP messages with both sites, it has the information identifying the correct gateway to use when reaching those sites.

5.2 ARPANET/Milnet Split

In October 1983, DARPA split the ARPANET into two physical networks, one for experimental research, and the other for production service. The new network, called MILNET, carried traffic between military installations needing reliable network service, while researchers continued using the ARPANET as an experimental network. Although the networks were physically distinct, both networks used the same hardware and software technology, and BBN operated them both. Both networks remained part of the same autonomous system, and seven special LSI-11 gateways called *mailbridges* forwarded traffic between the two networks.

The existence of multiple gateways between the two cross-country networks presented a new problem. Because IP knows only about networks, an ARPANET gateway on the east coast could not determine whether a destination on Milnet resided on the east or west coast. Clearly, an ARPANET gateway on the east coast would not want to forward traffic destined to an east coast MILNET site through a mailbridge located in California.

The Core system supported load-balancing of traffic between hosts on the two networks. BBN engineers performed extensive traffic analyses to determine how traffic should be divided among mailbridges and placed configuration tables in each Core gateway that identified which mailbridge should carry traffic between given source-destination pairs. The load-balancing mechanism worked only for datagrams carrying source and destination addresses on the ARPANET and Milnet networks. If an ARPANET host, for instance, forwarded a datagram destined for a host on Milnet to the "wrong" gateway, the gateway would send a redirect to the host identifying the correct mailbridge.

Because the load-balancing mechanism depended on a gateway's ability to determine which ARPANET or MILNET machine sent a datagram, it was ineffective at balancing traffic flows between sites not directly connected to the two networks. Indeed, as the Internet grew, the two networks carried an increasing amount of transit traffic between sites connected only indirectly to the networks. Instead, hosts connected to LANs which connected to a gateway on the ARPANET or Milnet.

After the split, ARPANET sites continued to run EGP with EGP servers connected to the ARPANET and similar servers were placed on MILNET. With the two networks, however, the extra-hop problem reappeared. In particular, mailbridges used GGP to obtain routing information, but did not understand EGP. Figure 5 illustrates the problem. Gateway G1 informs EGP server E1 of the networks within its autonomous system, and E1 advises G1 of the networks behind the mailbridges. The EGP servers, however, use GGP to propagate the information to the mailbridges. Although G1 forwards traffic for sites behind the ARPANET to the mailbridges directly, the

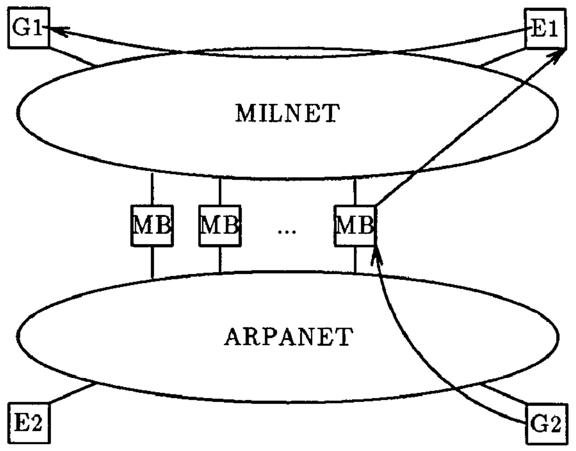

Figure 5: With the ARPANET/Milnet split, the extra-hop problem reappeared. Datagrams travelling from G2 to G1 follow the path G2-MB-E1-G1, needlessly visiting EGP server E1.

mailbridges forward traffic destined for sites behind G1 through the EGP servers. The extra-hop problem between the two networks remained until January, 1989, when the LSI-11 mailbridges were replaced with Butterfly gateways.

5.3 Phase 1 NSFnet (Fuzzballs)

In 1984, the National Science Foundation (NSF) embarked on a program to provide scientists with high-speed access to supercomputers. Although the primary goal focused on supercomputer access, NSF recognized that it could achieve the primary goal and at the same time provide the basis for a nationwide academic research network [JLF*86]. Rather than build a single new physical network, a key recommendation of the NSFnet program was to adopt Internet technology and build a collection of networks.

The first phase of the NSFnet program, started in June 1986, used 56 kbps lines to link Fuzzball gateways at six supercomputer centers. Additional networks, in the form of campus and regional networks, connected to the backbone at supercomputer centers. Despite its growth to over 200 networks, the NSFnet backbone, regional, and campus networks were administrated as a single autonomous system, and NSFnet's constituent networks exchanged routing information with an IGP rather than EGP.

In the backbone, Fuzzball gateways used the HELLO protocols to exchange routing information [MB87]. Because the HELLO protocol was not widely implemented, regional and campus networks were unable to use the protocol. Moreover, the diverse software and hardware technologies employed by campus and re-

RIP metric	Delay (ms)	RIP metric	Delay (ms)
0	0	9	2322
1	100	10	3440
2	148	11	5097
3	219	12	7552
4	325	13	11190
5	481	14	16579
6	713	15	24564
7	1057	16	30000
8	1567		

Figure 6: *Gated* translates HELLO metrics into the smallest RIP metric such that if the RIP metric is translated back to a HELLO metric, the translated HELLO metric is no smaller than the original.

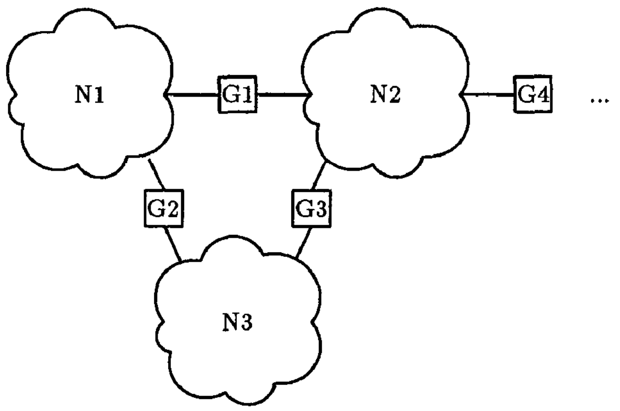

Figure 7: If gateway G1 readvertises routes learned from G4 at lower metrics than advertised by G4, loops can form when the information filters back to itself over the path through G2 and G3.

gional networks called for a widely available, standard protocol. RIP was the only available candidate.

Regional networks used Ethernets to connect to the backbone at supercomputer sites, and the regional gateways used the HELLO protocol to exchange routing information with the backbone Fuzzballs. A special routing daemon residing on the regional gateway, *gated* [Fed88], translated the HELLO-derived routes into RIP updates that it forwarded to gateways on the regional network. *Gated* translated the HELLO metrics of routes received from the Fuzzball into metrics usable within the context of RIP and translated RIP metrics to HELLO metrics in updates sent to the Fuzzballs.

Translating metrics proved to be difficult because HELLO metrics denote delays, while RIP metrics denote hop counts. *Gated* mapped RIP metrics into HELLO delays and vice versa using the rules given in Figure 6 [FH88].

NSFnet grew rapidly after its birth. Within two years, more networks were reachable through NSFnet than through the Core itself. As NSFnet grew, so did the number of interconnections between networks within NSFnet. For instance, some regional networks connected to the backbone at multiple points, and "back door" links connected some regional networks to others. Often, several paths existed between a given pair of hosts. Growth led to two problems. First, the distance between some networks exceeded RIPs notion of infinity. Second, the rich set of interconnections between constituent networks produced routing instabilities commonly associated with vector-distance protocols. The following subsections articulate the difficulties.

5.3.1 Exceeding RIP's Notion of Infinity

The sheer size of NSFnet exposed a significant problem: the combined diameter of the backbone, regional and campus networks exceeded sixteen network hops, RIP's definition of infinity. Although the value of the metric denoting infinity could have been raised, the idea was rejected for fear of exacerbating the counting-to-infinity problem. Instead, *gated* software was modified to selectively change the value of RIP metrics received in routing updates. A gateway receiving an update advertising a route at a metric of fourteen, for example, could readvertise the route at a metric of seven.

Allowing gateways to remap metrics in received routing updates presented NSFnet with a significant management problem because remapping metrics can introduce routing loops. The difficulty arises when a gateway advertising a network at the correct metric receives an update from a gateway advertising the network at a lower, remapped metric. The path advertised at the lower metric may, in fact, traverse the gateway receiving the update.

The simple topology in Figure 7 illustrates the problem. Gateway G4 advertises a path for some network N at metric 12, and gateway G1, changes the metric to 4 in the update it sends to G2. G2 advertises the route to G3 at metric 5, and G1 changes its routing table to use the path advertised by G3 at metric 5. Unfortunately, a routing loop has now developed between G3 and G1.

In the complex topology of NSFnet, choosing metrics and regulating which gateways could safely propagate updates about individual networks entailed significant administrative management. The task could not be automated and required tedious human oversight.

5.3.2 RIP Extensions

As NSFnet increased in size, routing instabilities became common. Sites complained that they could not reach some sites at all, while other sites would oscillate between being reachable and unreachable. Efforts aimed at stabilizing routing, met with only marginal success. Efforts included adding a hold-down timer to RIP software, adding split horizons, and finally split horizon with poison updates. Additional experiments investigated the effects of changing the mappings between RIP and HELLO metrics. Although some sites noted some improvements, the overwhelming evidence pointed to the need for a replacement routing technology. In short, the problems outlined in [MFR78] were rediscovered.

5.3.3 NSFnet and the Core

Before NSFnet, most sites connected to the ARPANET Core at a single point. The NSFnet backbone, however, connected to the ARPANET Core at several points. As with connections to regional networks, backbone Fuzzballs did not connect directly to the ARPANET; instead, they shared a common Ethernet with gateways running *gated* on the ARPANET. In order to distribute the traffic load among the ARPANET/NSFnet connections, NSFnet employed fallback routing [MB87]. Under normal circumstances, traffic travelled through primary gateways. If the primary gateway failed, however, traffic would be diverted to secondary, backup gateways. When the the primary gateway resumed proper functioning, traffic flows would revert to the primary gateway.

Each of NSFnet's constituent networks was assigned a primary gateway, and the others were dubbed secondary gateways. The primary gateway for a network advertised the network to the Core at an EGP metric of zero, while the secondary gateways advertised a network at a metric of three. Because gateways preferred paths advertised at a lower metric, traffic originating from the core would travel through a destination's primary gateway when it was available. Should the primary gateway fail, traffic would revert to the secondary gateways.

Although the backbone ran EGP with the Core, none of the routes learned from the Core were propagated to the Fuzzball backbone or its regional networks. Instead, NSFnet gateways running EGP with the ARPANET EGP servers advertised a wildcard route. Through the HELLO protocol, the Fuzzball backbone propagated the wildcard route to the regional networks. In turn, the regional networks propagated the wildcard route via RIP to end-user sites.

5.4 Phase 2 NSFnet (Merit)

Phase two of the NSFnet project became operational in June 1988, when all traffic on the Fuzzball-based NSFnet was diverted to a new backbone operated by Merit. The new backbone used T1 speed[5] links to interconnect packet switches called Nodal Switching Subsystems (NSS).

Based on the routing experiences of its predecessor, the new backbone adopted a significantly improved routing architecture [Bra88]. First, the backbone and each regional network operate as distinct autonomous systems, and they use EGP to exchange routing information with each other. Second, the backbone itself uses an SPF-based link-status IGP [Rek88a]. Finally, NSS nodes filter all EGP updates through a configuration database [Rek88b]. Information contained in updates, but not consistent with the database, is ignored and brought to the attention of a network manager.

Filtering updates through a configuration database introduced an administrative aspect to the backbone not present before. Specifically, the database describes the expected and permitted interconnections among networks. New networks that join the Internet must be added to the database before the backbone will propagate reachability information about them. Carefully controlling the propagation and use of route information, however, decreases the probability that loops will form.

5.5 Butterfly Mailbridges

In December of 1988, BBN replaced the aging LSI-11 mailbridges with six Butterfly gateways. In addition to routing traffic between the ARPANET and MILNET, the mailbridges assumed the roles of the EGP servers, eliminating the extra-hop problem between the ARPANET and MILNET.

Butterfly gateways use a link-status algorithm to distribute routing information about internal networks, those networks connected to Butterfly gateways. However, the link-status algorithm does not propagate routes for EGP-learned networks. To propagate EGP-learned routes to other Butterfly mailbridges, Butterfly gateways implement an ad hoc protocol called *Spread* [Atl89]. Spread is a modified implementation of EGP, in which gateways generate periodic, unsolicited update messages every sixty seconds. Moreover, Spread omits echo request and reply messages, and update messages include an autonomous system identifier of the gateway advertising the path.

[5] T1 speed links have a carrying capacity of 1.544 Mbps. At present, the NSFnet backbone links operate at 1/3 T1 capacity.

The Butterfly mailbridges also support a load-balancing mechanism effective for all IP-level traffic. While the load-balancing mechanism of the LSI-11 Core gateways relied on ICMP redirects, the Butterfly mailbridges use EGP to specify which mailbridges a particular gateway should use. When generating an update, the mailbridge examines the IP address of the EGP neighbor targeted by the update and uses the address to determine which mailbridge the target gateway should use to reach sites on or behind the other network. For example, if a non-Core gateway on the east coast peered with an ARPANET mailbridge on the west coast, the EGP updates generated by the west coast mailbridge would identify an east-coast mailbridge as the correct gateway to use when sending traffic to sites on or behind Milnet.

6 Conclusions

Explosive growth is taxing current Internet routing mechanisms. New sites continue to join the Internet on a daily basis, and sites add new links to destinations with which they desire better connectivity. In some sense, the Internet is a victim of its own success; many routing protocols are being used in environments for which they had not been designed. For instance, the EGP model disallows the existence of multiple paths between sites in different autonomous systems, forcing sites to use ad hoc techniques such as running IGPs between sites in different autonomous systems. Moreover, the Core model along with the absence of a universal EGP metric definition prevents autonomous systems from interconnecting in convenient ways, restricting the visible topology of the Internet to a tree. To keep routing stable, the new NSFnet backbone has resorted to the use of a centralized routing database that restricts the allowable Internet topology. There is a critical need to explore alternative models such as [Mil86] that loosen the topological restrictions on Internet interconnections to reflect the extent and manner in which sites actually connect to one another.

References

[Atl89] Steve Atlas. Bolt, Beranek, and Newman Inc., Private communication, February 1989.

[BG87] Dimitri Bertsekas and Robert Gallager. *Data Networks.* Prentice Hall, 1987.

[Bra88] Hans-Werner Braun. The NSFNET Routing Architecture. DARPA Networking Information Center, RFC 1093, May 1988.

[BSTM80] David R. Boggs, John F. Schoch, Edward A. Taft, and Robert M. Metcalfe. Pup: an Internetwork Architecture. *IEEE Transactions on Computers*, COM–28(4):612–623, April 1980.

[Com88] Douglas E. Comer. *Internetworking With TCP/IP: Principles, Protocols, Architecture.* Prentice Hall, 1988.

[Fed88] Mark S. Fedor. GATED: A Multi-Routing Protocol Daemon for UNIX. In *Proceedings of the 1988 Summer USENIX Conference*, pages 20–24, Winter 1988.

[FH88] Mark Fedor and Jefrey C. Honig. *Man page for gated.* Cornell Theory Center, Cornell University, Ithaca NY 14853-5201, May 1988.

[Hed88] Charles Hedrick. Routing Information Protocol. DARPA Networking Information Center, RFC 1058, June 1988.

[HHS83] Robert Hinden, Jack Haverty, and Alan Sheltzer. The DARPA Internet: interconnecting heterogeneous computer networks with gateways. *Computer*, 16(9):38–48, September 1983.

[HS82] Robert Hinden and Alan Sheltzer. DARPA Internet Gateway. DARPA Networking Information Center, RFC 823, September 1982.

[JBH78] Irwin Mark Jacobs, Richard Binder, and Estil V. Hoversten. General Purpose Packet Satellite Networks. *Proceedings of the IEEE*, 66(11):1448–1467, November 1978.

[JLF*86] Dennis M Jennings, Lawrence H. Landweber, Ira H. Fuchs, David J. Farber, and W. Richards Adrion. Computer Networking for Scientists. *Science*; 231:943–950, February 1986.

[Kir84] Paul Kirton. EGP Gateway under Berkeley UNIX 4.2. DARPA Networking Information Center, RFC 911, August 1984.

[MB87] David L. Mills and Hans-Werner Braun. The NSFNET Backbone Network. In *Proceedings of the ACM SIGCOMM '87 Workshop*, pages 191–196, August 1987.

[MFR78] John M. McQuillan, Gilbert Falk, and Ira Richer. A Review of the Development and Performance of the ARPANET Routing Algorithm. *IEEE Transactions on Computers*, COM-26(12):1082–1811, December 1978.

[Mil83] David L. Mills. DCN local-network protocols. DARPA Networking Information Center, RFC 891, December 1983.

[Mil84] David L. Mills. Exterior Gateway Protocol Formal Specification. DARPA Networking Information Center, RFC 904, April 1984.

[Mil86] David L. Mills. Autonomous Conferations. DARPA Networking Information Center, RFC 975, February 1986.

[MRR78] John M. McQuillan, Ira Richer, and Eric C. Rosen. *ARPANET Routing Algorithm Improvements First Semiannual Technical Report.* Technical Report 3803, Bolt Beranek and Newman Inc., April 1978.

[MRR80] John M. McQuillan, Ira Richer, and Eric C. Rosen. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Computers*, COM-28(5):711–719, May 1980.

[MW77] John M. McQuillen and D. C. Walden. The ARPANET Design Decisions. *Computer Networks*, 1(5), September 1977.

[Pos80] Jonathon Postel. Internetwork Protocol Approaches. *IEEE Transactions on Computers*, COM-28(4):605–611, April 1980.

[Pos81] Jonathon B. Postel. Internet Control Message Protocol. DARPA Networking Information Center, RFC 792, September 1981.

[QSP85] John S. Quarterman, Abraham Silberschatz, and James L. Peterson. 4.2BSD and 4.3BSD as Examples of the UNIX Operating System. *ACM Computing Surveys*, 17(4):379–418, December 1985.

[Rek88a] J Rekhter. EGP and Policy Based Routing in the New NSFNET Backbone. DARPA Networking Information Center, RFC 1092, March 1988.

[Rek88b] J Rekhter. The NSFNET Backbone SPF based Interior Gateway Protocol. DARPA Networking Information Center, RFC 1058, June 1988.

A Problems with Growth

This section chronicles significant events that lead to a break down of routing on an Internet-wide basis. As might be expected, many of the problems related to growth and occurred during the rapid growth of NSFnet.

Feb 1986 EGP servers began generating EGP updates exceeding 576 bytes in size. Many implementations of EGP, (e.g. Kirton's implementation for UNIX systems [Kir84]) allocated a static buffer of 576 bytes to hold incoming update messages. When an update exceeding 576 bytes in size was recieved, network software would truncate the message before handing it to the application, and EGP software would record a checksum error as it performed its integrity check. Assuming that the message was corrupted rather than truncated, EGP software discarded the entire update. Because the size of routing updates varied frequently from one update to another, a site would typically lose some updates, while correctly processing others. Internet routing became unstable with paths to destination oscillating between being reachable and unreachable. The problem of underestimating the size of EGP messages reappeared numerous times; by December, 1989, servers generated update messages exceeding 2048 bytes.

July, 1986 The number of networks in the Core exceeded 120 for the first time, overflowing the LSI-11 Core gateway routing tables. If a site's gateway crashed, EGP servers would free the table slots allocated to the crashed gateway, allowing them to be assigned to other gateways. Following a crash it might take days for a site's networks to reenter tables in every Core gateway. Not surprisingly, the problem reoccurred several times. By January, 1989, the Core contained routes to over 630 IP networks.

Likewise, the EGP servers could support only a limited number of EGP neighbors. Once a server reached its EGP neighbor limit, it would reject further peer requests from other gateways. If a site's gateway rebooted, it might lose its neighbor slot and find that no server would peer with it. Until a site was able to find a server that it could peer with, it was effectively cut off from the Internet.

Fall 1987 As the size of EGP updates and the number of gateways peering with EGP servers increased, the servers consumed more and more of their processing capacity handling routing updates. By the early fall of 1987, server CPUs were running at close to 100 percent capacity, and could no longer process updates promptly. Delays became so large that gateways would declare their EGP peers down. Internet routing essentially collapsed.

At an October 1987 meeting of the Internet Engineering Task Force (IETF) it was suggested that additional processing power could be realized by upgrading the LSI-11 Core servers to LSI-11/73 processors. The IETF promoted an "Adopt-a-Mailbridge Foster Parent Program" to search for spare 11/73 processors that could be temporarily loaned to the Core. One month later, all EGP servers and mailbridges had been upgraded, and Internet routing recovered.

September 1987 The size of EGP updates began exceeding 1012 bytes, the maximum size datagram carried by the ARPANET and Milnet. Moreover, the Core gateways did not fragment or reassemble IP datagrams, truncating large messages instead. Users encountered routing difficulties with symptoms analogous to the those that occurred when the EGP servers overflowed their internal route tables. Six weeks elapsed before EGP server software began fragmenting and reassembling large updates.