The Birth of Link-State Routing

John M. McQuillan *Editor: David Walden*

''Routing is a hard problem.'' That's what colleagues told me when I joined BBN in 1971, right after getting my undergraduate and master's degrees in computer science at Harvard University. As a student, I had built the hardware and software to connect Harvard's first interactive computer, the DEC PDP-1, to the infant Arpanet (the first packet switching computer network and precursor of the Internet).[1] I had also taken a Harvard course taught by two senior BBN engineers who built the original interface message processors (IMPs), the switching nodes that routed traffic across the network. While at BBN, and with almost all the course requirements for a PhD in hand, it was time to find a possible dissertation topic. So I was glad to learn that dynamic routing—determining the best paths for network traffic in real time—was considered a difficult computer science issue because I planned to tackle it.

By 1971, the Arpanet had been up and running long enough to disclose shortcomings in the performance and stability of the original design. My job at BBN was to redesign and rewrite all the software for the IMP, not just the routing module. I fondly recall releasing new IMP software to the whole network (amounting to a few dozen nodes at the time) every other Tuesday morning for two years. Fifty software versions later, the IMPs had more stable congestion management, better reliability, and higher throughput.[2] But routing still remained a challenging area for further study.

By the end of 1974 I had completed my PhD dissertation for Harvard on routing[3] while working full-time at BBN. This dissertation described the problem, analyzed and compared many routing algorithms, and pointed out topics for further work, such as hierarchical routing for networks of networks. But I had not resolved some of the nagging network accidents, outages, and other crises caused by the original Arpanet routing system.

Why routing is a difficult problem

Routing is challenging for three reasons. It's a real-time software process running on many nodes.[4] Second, it's a significant optimality problem when considered as an algorithm running on a single node. Finally, it's demanding of system resources, placing further constraints on what is possible in practice.

Routing in a distributed computer network (the Arpanet being the prototypical example) is unlike most other software applications:

- Routing is a nonstop application; it does not start with a set of known initial conditions nor does it run to completion.
- Routing is decentralized, with no master clock, and no master routing table. Each node must perform its own routing calculations.
- The routing process is inherently distributed. It runs in all nodes all the time. The constraints of time and space imply that nodes are working with different information. Routing loops are a snag to be avoided.
- Routing must be adaptive, using alternate routes to avoid congested parts of the net.
- Routing continuously measures the net, while sending traffic over it, affecting the measurements. Oscillations are a danger; stability is a challenge.
- Routing must also be a fail-safe application that continues to function if nodes fail, even if the net is partitioned into two or more pieces. Each subset of the network should keep working, and recover seamlessly when an outage is repaired.
- Perhaps the scariest aspect of routing is that it is mission-critical to the entire network. As we had ample opportunity at BBN to witness, failures in routing can cause an entire network to stop working. For instance, in 1971 the IMP at Harvard had a memory error that caused its routing updates to be all zeroes. That led all other nodes in the net to conclude that the Harvard IMP was the best route to everywhere, and to send all their traffic there. Naturally, the network collapsed. We developed heuristics to detect such problems and prevent their wider propagation. The fact remains that local routing failures, unlike almost any other local failure, can have global consequences.

When considered as a classic algorithm, routing is an ''interesting'' computer science problem, with three critical, interrelated components:

- As a real-time application, the algorithm must include a measurement process to determine pertinent network characteristics, such as the current connectivity, traffic levels, delays, and available capacity. This is not clear-cut.
- The algorithm must include an efficient, reliable protocol for exchanging this information with other nodes. Failures in the exchange protocol can have drastic consequences.
- At the heart of the algorithm is the calculation of optimal routes over which network traffic should flow based on the characteristics measured and then exchanged with other nodes. Routing should achieve a reasonable level of optimality in a reasonable amount of time, and be stable in conditions of stable traffic.

Published by the IEEE Computer Society 1058-6180/09/$25.00 © 2009 IEEE

Finally, routing can consume a lot of system resources, such as memory, transmission capacity, and processing capacity. Any budget on the overhead devoted to routing (e.g., no more than ×% of transmission capacity) directly affects how adaptive the algorithm can be. Optimality is traded against cost—the key measure is the cost to perform routing as a function of network size.

Limitations of distance-vector algorithms

The original Arpanet routing algorithm[1] was a breakthrough when compared with fixed, centralized routing systems as practiced in 1960s telephony. In this prototypical distance-vector routing algorithm, each IMP estimated the minimum delay to each possible destination in the network. It periodically exchanged these estimates with the adjacent IMPs. Each IMP then recalculated the minimum delay to each destination based on this new information, using the adjacent IMP with the best path in each case.[5] The three components of distributed adaptive routing were in place: measurement (in this case the length of packet queues to each destination); protocol between nodes to share the results; and calculation based on the optimality principle that optimal paths are made up of optimal subpaths.

This routing procedure served quite well for the Arpanet's first five years or so—its most active research phase as a prototyping facility. But as the network grew, and more universities and military facilities began to rely on the network for doing ''real work,'' pressures built for the research community to solve some fundamental weaknesses of the distance-vector approach.

This algorithm reacted quickly to good news but slowly to bad news. If one node reported a path with a better delay, all nodes learned it quickly. But if the delay increased, the nodes did not use the new information while they still had adjacent nodes with old, lower values. The algorithm was designed for a small homogeneous network. The Arpanet had grown significantly, and included lines and nodes of different capacities and delay characteristics. By 1977, our studies for ARPA had made it clear that the routing algorithm was performing slowly and poorly under heavy load and congestion was leading to troublesome network disturbances.[6] Finally, its resource utilization scaled linearly with the network size, so routing was becoming too expensive. The Arpanet had outgrown its original routing algorithm.

Development of link-state routing

In August 1977, I led a BBN team that submitted a proposal to ARPA to investigate these and other shortcomings and to develop a better solution.[7] The proposal was accepted, and we spent the next 18 months developing and implementing the first link-state routing algorithm, which we called Shortest Path First (SPF). Ira Richer and Eric Rosen, two of my colleagues in the initial research, made major contributions to SPF. I was exceptionally fortunate to win a sizable grant for what amounted to postdoctoral research, to work with a top-notch team, and to have direct ongoing access to the world's finest research network as a testbed.

By this time, I had been thinking about routing for seven years. Wouldn't it be better, I wondered, if every node in the network had the entire network ''map,'' instead of routing tables from adjacent nodes? If every node had current traffic reports, and could do the entire routing calculation independently, that might eliminate the instability and congestion of distance-vector routing. A local (rather than distributed) calculation could solve many problems. Ideally, every network node would have complete information about all the links in the network. This would involve a paradigm shift—from an exchange of *path* information to an exchange of *link* information.

But there was a show-stopper: for a long time no one believed that link-state routing was feasible. We assumed we couldn't afford the resource cost of informing every node about every change in the delay of every link. We also assumed we couldn't afford the processing time in each node to react to each change, to calculate optimal paths to every network node. We didn't believe such a solution would work in large-scale networks. After all, scalability was one of the key problems we were already having with the original, simpler, algorithm.

So the invention of link-state routing was born out of the necessity to replace distance-vector routing, even though we weren't sure at first that it would be practical. When we wrote the ARPA proposal, I was excited about developing a link-state solution, but that wasn't the only idea we proposed to examine. I was not fully confident we could deliver an algorithm to meet all the constraints. This was research, without guaranteed results. In the end, we were pleased to achieve a working link-state routing implementation in the Arpanet, a classic use of the research network for proving a new concept and validating its feasibility.

Three engineering innovations were needed to develop scalable link-state routing. First, we changed the delay measurement to record expected, not instantaneous, delay. The old algorithm updated its estimates eight times a second. By spring 1978, we'd completed extensive measurements on the Arpanet and concluded that a 10-second interval was optimal for estimating expected delay. We then ran a median smoothing algorithm on the raw data to filter out noise. We had better, more stable data for the routing algorithm, delivered less often, only when a meaningful change occurred.

The next innovation was to improve the updating protocol, replacing periodic with event-driven updating of delay tables. We analyzed ways to propagate changes in network delay. A crucial step was discovering the advantages of ''flooding,'' in which each node sends each new update on all its lines except the line on which it was received. I was elated to perform the calculations and realize that flooding would cost only 90 bits per second in a 100-node net, and 1,000 bits/sec in a 1,000-node net (as far as our imagination could reach in those days). A key advantage of flooding information about links was that it isolated the updating protocol from the routing calculation, which improved reliability and stability. The updates did not need to be processed, and were small, so they could flow quickly, enabling more adaptive routing. Achieving better optimality at lower cost seemed almost too good to be true.

The third innovation, and the one that gave our algorithm its name, was the use of Dijkstra's shortest-path-first algorithm[8] for calculating a tree of optimal routes from a database of link delays. As a distributed adaptive algorithm, SPF has an enormous advantage over Dijkstra's algorithm. At all times, the node already has the tree of optimal paths. When a delay update comes in, it's not necessary to recalculate the entire tree. The running time of the incremental version of SPF is proportional to the number of nodes affected by a link change; SPF merely computes the new subtree below the affected link. I showed that the average subtree size in a network is equal to the average path length, which is highly significant because average path length grows very slowly with the network size. For example, in a network with each node having three adjacent nodes, a thousand-node network has an average path length of only 8; a 10,000-node network would have a length of only 11. It was a key conceptual breakthrough to realize that SPF can calculate a changed routing tree in a fraction of the time it would take to calculate a whole new tree; speedup is 100-fold in the thousand-node example. This made the link-state calculation feasible.

Not only was link-state routing feasible, it was exceptionally efficient, with transmission and processing overhead of less than 2% in the Arpanet of the late 1970s, less than half the cost of the original approach. In addition, in an era when memory was far more expensive than it is today, we fit the topology database, and the entire algorithm—-measurement, protocol, and calculation packages—into 4 Kbytes of memory.

Almost unbelievably, it began to appear that we had finally built a scalable link-state algorithm. The original protocol scaled linearly with network size. We showed that the transmission cost of the new flooding protocol scaled logarithmically with network size, and that the processing time of the calculation for each update grew logarithmically.

The final step was to make sure it actually worked under the Arpanet's operational conditions. We installed SPF in the Arpanet in a carefully controlled series of releases during late 1978 and 1979. James Herman made key contributions to this phase, in addition to the original team members. The Arpanet offered us the best of both worlds—a research environment that was also a large network with heavy real-world traffic. We tested and measured for many months. SPF reacted to most changes within one-tenth of a second throughout the network. It found shortest paths, adapted to outages, and avoided congestion in most cases. Oscillations were minimal. We had met our objectives—success!

ARPA encouraged us to publish our work without any restrictions.[9,10] Our results were cited widely.

Link-state routing after Arpanet

This project was my last research effort. Although I found it rewarding to be involved in the creation of link-state routing, it was less rewarding to be involved in other aspects of research. Disputations with the academic world came with the territory. Many people thought they could do a better job than we had. No doubt some could have, but it was irksome to be in the situation of comparing our implementation, actually running in a large network, with any number of hypothetical alternatives. My frustrations presaged the later insistence on ''running code'' in Internet engineering circles.[11]

I left BBN in 1982 after a deeply satisfying decade there, and worked as an independent network consultant, conference organizer, and investor for the next 20 years. I gave advice to carriers, vendors, users, venture capitalists, and startups. During this later stage of my career, I noted the gradual diffusion of link-state routing into the Internet, many years after its success in the Arpanet:

One decade later—standardization. I was flabbergasted that Internet gateways and routers were using distance-vector routing well into the 1990s. Why would anyone accept the limitations of distance-vector routing, after we had shown its many flaws and pointed out a better way? (The various reasons need not detain us here). Eventually, in 1989, a decade after SPF, link-state routing became the subject of international standardization. OSPF[12] and IS-IS[13] were specified for TCP/IP and OSI, respectively. These first-generation descendants of SPF have been followed by several subsequent generations of standards, right up to the present.

Two decades later—commercialization. My Arpanet perspective gave me a head start when the Internet evolved from government sponsorship to Wall Street backing. In 1995, I retired from consulting and began working as an independent venture capitalist. During the extraordinary gold rush to construct the broadband Internet, I invested in several dozen network startups and worked closely with the entrepreneurs. Routing was recognized as a critical success factor by vendors throughout the value chain, from chips to systems to services, and from local nets to backbone routers. Over a dozen of the startups I supported developed routing technology. These SPF descendants were implemented in silicon, running literally a million times faster than 20 years before, with gigantic routing tables. We can be proud of this further demonstration of scalability.

Three decades later—mass commoditization. Today, link-state routing is one of the Internet's building blocks. It's rewarding to see our early feasibility work bear fruit on a scale we never could have imagined. Annual sales of routers and switches now exceed $10 billion.

In my opinion, ARPA made the world's best investment in technology when they funded the Arpanet and anticipated the Internet. It was great to be a part of it.

References and notes

1. W.R. Crowther et al., ''The Interface Message Processor for the ARPA Computer Network,'' *Proc. 1970 Spring Joint Computer Conf.,* AFIPS Press, 1970. The Advanced Research Projects Agency (ARPA) of the Department of Defense awarded BBN the contract to build the Arpanet in 1969.
2. J.M. McQuillan et al., ''Improvements in the Design and Performance of the ARPA Network,'' *Proc. 1972 Fall Joint Computer Conf.,* AFIPS Press, 1972.
3. J.M. McQuillan, ''Adaptive Routing Algorithms for Distributed Computer Networks,'' BBN Report No. 2831, May 1974 (dissertation).
4. I use ''node'' here to refer to any system that performs routing, starting with the original Arpanet IMPs, and continuing on to today's ''routers,'' with many intermediate terminological generations such as ''gateways.''
5. See, for example, http://www.walden-family.com/public/bf-history.pdf.
6. J.M. McQuillan, G. Falk, and I. Richer, ''A Review of the Development and Performance of the Arpanet Routing Algorithm,'' *IEEE Trans. Comm.,* vol. COM-26, no. 12, Dec. 1978.
7. ''Arpanet Routing Algorithm Improvements,'' BBN Proposal P77-CSY-93A, 4 Aug. 1977.
8. E. Dijkstra, ''A Note of Two Problems in Connection with Graphs,'' *Numerische Mathematik,* vol. 1, 1959, pp. 269-271.
9. J.M. McQuillan, I. Richer, and E.C. Rosen, ''An Overview of the New Routing Algorithm for the Arpanet,'' *Proc. Sixth Data Comm. Symp.,* 1979; also *ACM SIGCOMM Computer Comm. Rev.,* vol. 25, no. 1, Jan. 1995, 25th anniversary issue ''Highlights from 25 years of the Computer Communications Review.''
10. J.M. McQuillan, I. Richer, and E.C. Rosen, ''The New Routing Algorithm for the Arpanet,'' *IEEE Trans. Comm.,* vol. COM-28, no. 5, May 1980.
11. A.L. Russell, '''Rough Consensus and Running Code' and the Internet-OSI Standards War,'' *IEEE Annals of the History of Computing,* vol. 28, no. 3, 2006, pp. 48-61.
12. J. Moy, ''The OSPF Specification,'' NWG RFC 1131, Oct. 1989.
13. D. Oran, ''OSI IS-IS Intra-domain Routing Protocol,'' NWG RFC 1142, Feb. 1990.

Contact John McQuillen at john@mcquillan.com. Contact department editor David Walden at annals-anecdotes@computer.org.