How we made the modern Ethernet

The spanning tree protocol is a key component of today's Ethernet. Radia Perlman, inventor of the underlying algorithm, recounts how it was first developed.

Radia Perlman

Ethernet today is completely different from the original Ethernet. The original, which was conceived by researchers, including Robert Metcalfe and David Boggs, at Xerox PARC in the early 1970s, was a single wire shared by a few hundred computers. Since data could not be received properly if there were a collision (multiple computers transmitting simultaneously), the key concept in the original Ethernet involved how to avoid collisions. Technologies that are limited to a single link are known as local area networks (LANs). Larger networks (such as the Internet) need computers known as routers, which forward data between links.

In the early 1980s, a few years before Ethernet was commercially deployed, I was at Digital Equipment Corporation, responsible for designing the component of a network that allows a router to receive a data packet addressed to a destination, and forward the packet toward that destination. Each router participates in a routing protocol with other routers, where they exchange information in order to compute a forwarding table. The forwarding table tells the router how to send data on the path to the destination.

Then along came Ethernet. My routing protocol design assumed links were between just two computers (point-to-point links), but Ethernet was a wire with hundreds of computers that could all directly hear each other, and thus were all neighbours. I made modifications to the routing protocol to efficiently accommodate this new type of link. However, perhaps because Ethernet was called Ethernet and not Etherlink, people built applications that would just work on a single Ethernet link, leaving out the information and protocol necessary to allow a router to forward data between links. I tried to argue that the applications should be using a network protocol (such as IP), rather than simply Ethernet, because the application might need to work between computers on different links. "Our customers would never want to do that," was the typical reply.

But people did, of course, want applications to work on larger networks

than a single Ethernet. In 1983, my manager, Tony Lauck, asked me about designing a 'magic box' that would let computers on separate Ethernets communicate (this box would become known as a bridge or switch). This was exactly what the network layer was for, but the network layer requires every computer attached to the network to support the network layer, and people had built applications that only supported Ethernet. My first suggestion was that people should fix the applications. But Tony said that we needed a quick fix until there was a chance to change the software in all the attached computers.

One suggested approach was to have a bridge simply listen to every packet on each of its links, and forward it to each other link. The problem with this was that if there were loops (multiple paths between the same pair of computers), Ethernet packets would just keep circulating. Forbidding loops had some severe limitations. First, a customer might accidentally plug together devices with a loop, and break the entire network. Second, loops enable alternative paths in case a bridge or link fails. So rather than simply forbid loops, Tony challenged me to allow loops, but invent a mechanism for bridges that finds a loop-free subset of the topology that interconnects all the computers. As the topology changes due to bridges or links being added to the network, failing or recovering, the mechanism should adjust the data paths as necessary, to always maintain a loop-free path for data. Tony thought it would be a difficult problem, perhaps even impossible, so just to be funny, he threw in an extra challenge: to make the amount of resources (memory, for example) necessary to run the algorithm not grow as links and bridges were added to the network.

It was a Friday afternoon when Tony first posed this challenge, and he was going to be away and unreachable the following week. (This was a time before mobile phones or reading e-mails while away from the office.) That night I realized exactly how to solve the problem. The basic idea was for the bridges to collectively select a single bridge, calculate a tree with that bridge as the root

Fig. 1 | Topology of a spanning tree network.

Circles represent bridges, solid lines represent links and squares represent computers. The dashed lines indicate links that are not in the chosen spanning tree.

(a spanning tree), and only forward data packets to and from links in the tree (Fig. 1). The amount of resources needed to run this algorithm would also not depend on how many links and bridges there were.

By Tuesday afternoon, I had written the specification, which proved to be sufficiently clear and complete that the bridge implementers at Digital Equipment Corporation were able to implement it within a couple of months without asking me a single question. But I couldn't show it off to Tony until he returned the following week, and was too excited to work on anything else, so I spent the remainder of the week working on a poem (called *Algorhyme*) that would become part of the abstract of the [paper](#) in which I published the spanning tree algorithm.

Spanning tree bridges quickly replaced the original concept of Ethernet. Today's Ethernet does not involve shared links, but rather consists solely of point-to-point links, with bridges running the spanning tree algorithm. This spanning-tree-based design is widely deployed today, and has changed very little in the past 35 years. □

Radia Perlman
Dell EMC, Seattle, WA, USA.
e-mail: radia@alum.mit.edu

Published online: 12 October 2018
<https://doi.org/10.1038/s41928-018-0153-6>