from *SIAM News*, Volume 35, Number 4

Beyond Reed–Solomon: New Codes for Internet Multicasting Drive Silicon Valley Start-up

By Sara Robinson

In 1996, Michael Luby was ready to trade his academic hat for a practitioner's one. He was tiring of the academic cycle of proving theorems and writing papers from his post as head of the theory group at the International Computer Science Institute in Berkeley. Moreover, with the technology revolution ramping up, the time seemed right to launch a start-up.

Luby and some of his theory group colleagues had already been thinking about ways to apply their theoretical knowledge of coding theory to various practical problems of Internet data transmission. The challenge they settled on was the problem of efficiently and reliably sending data from one computer to a vast audience, perhaps thousands or millions of recipients.

The Multicast Problem

Distributing a digital copy of a movie or a software package to a large number of people is hard to do efficiently with the protocols used for e-mail or Web pages. When a computer sends a data file to a single recipient on the Internet, it typically uses the Transmission Control Protocol (TCP), which breaks the information into chunks, called packets, that then whiz along possibly separate routes to the receiving computer. For each packet received, an acknowledgment is sent back to the sender. If a packet is sent and not acknowledged after a window of time, the sending computer will automatically resend the packet.

When the same data file is sent to many recipients, bandwidth and server costs are dramatically reduced if the sender doesn't maintain a separate connection with each end recipient but rather "multicasts" the data. That is, it sends only one stream of packets, which are replicated at routers and then fan out in a tree pattern toward the recipients. This protocol plays havoc with the TCP acknowledgment system, however, as the flood of acknowledgments can overwhelm the server. And a packet missed by one recipient will have to be resent to all of them, counteracting the benefits of multicast.

Thus, an open problem for multicast was reliability—how to ensure that every packet sent would actually arrive at its destination. One potential solution to the problem was to make use of coding theory. If redundancy is added to the original data to make up for packet losses, then there is no need for acknowledgments.

But there was a fundamental problem on the coding side as well. Coding theory was already being used for streaming media applications and satellite transmissions. But the Reed–Solomon codes—the industry standard—are optimized for the problem of correcting errors, as when a bit is flipped from 0 to 1, and not for the multicast problem, where some bits don't show up at all (although it is known which bits have been lost). As the size of a file grows, the encoding and decoding speed for Reed–Solomon codes, quadratic in the number of bits, becomes terrible. Thus, a data file must be broken into small blocks and each encoded separately, with the pieces interleaved to protect against small bursts of data losses. Still, random data losses can wipe out enough of a block of data to make it unrecoverable; Reed–Solomon codes, then, do not solve the multicast problem.

Luby's idea was to devise a new type of code, one that would better address the multicast problem. "People were trying to solve the network scalability problem directly," he says. "What I realized is that if you could solve the coding problem so that the codes scale with the size of the content, this would very neatly solve the network problem." Luby and his cohorts set out to develop codes that would work much faster than Reed–Solomon codes and protect against data loss rather than against errors. Together, they developed several iterations of codes, culminating in 1996 with a result they called "Tornado codes."

Two New Generations of Codes

Tornado codes, based on an approach pioneered in the 1960s by Robert G. Gallager, then a graduate student at MIT, use sparse, random, bipartite graphs. The codes correct erasures or missing bits, rather than errors, and they have linear-time encoding and decoding algorithms. Because of the speed of the encoding and decoding, an entire file can be encoded in one block, rather than being divided into smaller blocks. Once a sufficient number of encoded packets has been received, the recipient can reconstitute the file. Better still, for erasures, Tornado codes achieve the theoretical limits for efficiency, meaning that they are minimally redundant.

"Tornado codes were interesting both from a theoretical research standpoint and a practical one," says Amin Shokrollahi, a mathematician who specializes in coding theory and one of the developers of Tornado codes.

Armed with Tornado codes, Luby began meeting with venture capitalists and business advisers, and doing the paperwork involved in running a business. Meanwhile, the coding theory research continued.

Tornado codes, while a big step in the right direction, still had one major limitation: As with Reed–Solomon codes, devising an appropriate Tornado code for a given application requires that the researchers estimate, in advance, the loss probability of the channel to determine the amount of redundancy to put into the data. The measure of the redundancy is the "rate," the ratio of the number of information bits to the number of code bits for each block of data. A code with a rate of R can tolerate up to $1 - R$ errors. For most Internet applications, the loss probability varies widely over time. Thus, in practice, the loss rate had to be overestimated, making the codes slower to decode and far less efficient in transmitting information.

"With Tornado codes, you need to know loss probability on the channel, which you usually don't know—it's a guess," Shokrollahi says. "If you overestimate the loss probability, you waste bandwidth and if you underestimate it, the codes fail."

Luby's answer to these issues was a new type of code, now known as Luby transform codes. Devised in the summer of 1998, just after the company was founded, the new codes are similar in construction and properties to Tornado codes. Unlike other codes, however, Luby codes do not have a rate, because there is no notion of block length. Rather, the time it takes to recover the data is just the time needed to receive enough encoded packets. The encoded packets, called "metapackets," are generated in a probabilistic experiment performed on the original file. As with Tornado codes, once a recipient has obtained a sufficient number of these packets, the file can be decoded.

"Tornado codes stretch a fixed-length message to a fixed-length encoded message," Shokrollahi explains. "Luby codes have no fixed encoding rate and no rate. You can generate encoded packets on the fly and independently." With Tornado codes, he adds, generating a steady stream of packets means repeated cycling through a set of encoded packets. With Luby codes, by contrast, each packet generated is independently useful.

Luby Transform Codes: The Basic Algorithms

The encoding algorithm uses a weight table that gives a probability distribution on input symbols $1, \ldots, n$, which can be thought of as packets. For $k \geq 2$, the probability of weight k is about $1/k(k - 1)$ and the probability of weight 1 is about $1/n$.

To create a metacontent packet, a weight k is first chosen at random according to the distribution, and then k packets are chosen uniformly at random and XORed (added bitwise modulo two). The header for each metacontent packet contains information about which original data packets are included.

The receiver collects sufficiently many metacontent packets (about 105% of the original file). Because each metacontent packet knows which packets it includes, the client can set up the induced graph between original and metacontent packets, where an edge represents inclusion of that packet in the metapacket. The decoding algorithm starts by finding a packet of degree one. Because such a metapacket is just a copy of one of the original packets, the algorithm copies the value of that packet and XORs it out of the payloads of the remaining metapackets, creating new packets of degree one, and so on. Decoding, which is essentially the inversion of a special kind of random sparse matrix, is successful if all packets are connected to one metapacket and at each iteration there is at least one metapacket of degree one.

Thus, the key to the design of the Luby Transform codes is the weight table and the proof that, with this weight table, the probability of successful decoding is very, very high. Because the weights are, on average, small, the algorithm is also extremely fast.

"The smarts are in the analysis, not the algorithm," says Shokrollahi. "The engine is the weight table."—*SR*

An Unexpected Niche

In 1998, Luby launched the company, together with a Stanford business school graduate, Jay Goldin. The company of two was initially based in Oakland, California. Luby named the company Digital Fountain, a name selected for its vivid portrayal of what the technology supplies: a continuous fountain of packets that the intended recipients can dip into at any time and obtain a complete copy of the data. Luby initially envisioned the technology as a tool for downloading popular software packages, such as Adobe Acrobat, and indeed, Adobe was an early investor, as were Cisco Systems and Sony Corporation. Now, Cisco resells one of the Digital Fountain products integrated into a complete Cisco package, and Sony uses the technology to sell Digital Fountain products in Japan under the Sony name as well as to distribute data internally.

Over the next fall and winter, Luby and Goldin hired a number of engineers who began to implement their algorithms. The following year, Cliff Meltzer, a former Cisco executive, became the company's chief executive, while Luby retained the title of chief technologist. After exploring the market, the company executives decided on a business model of selling servers that implement Digital Fountain's algorithms.

As the engineers built the servers and the sales people went out to sell them, the company expanded, eventually outgrowing its Oakland offices and moving to San Francisco and, later, to Fremont, on the outskirts of Silicon Valley. Shok-rollahi, who had left ICSI for a position at Lucent Technologies' Bell Labs, returned to California in 2000 to become chief scientist of the company, which now has about 70 employees.

"At Digital Fountain, we are doing much more practical coding theory because it's driven by need," Shokrollahi says. "It's very problem-oriented, very, very focused research." By contrast, at Bell Labs, "you work in something for a little while and then you switch to something else."

Digital Fountain's research arm, consisting of Luby, Shokrollahi, and two others, has been working on a new generation of codes; the group is also addressing other related research issues, such as application of the technology to video-on-demand and streaming media. Recently, the group has been focused on congestion controls to ensure that the Digital Fountain packets behave in ways that don't cause network flow problems.

Some of Luby's friends from academia are consulting for Digital Fountain on the side. Richard Karp, of the University of California, Berkeley, and ICSI, has been working with Shokrollahi on the next generation of codes; Avi Wigderson of Hebrew University in Jerusalem and Hugo Krawczyk of the Technion in Haifa have been addressing security issues, such as a packet-authentication system for verifying that data have come from a particular source.

Launched to address the problems of streaming media and multicast, Digital Fountain—like many new technology companies—has found an unexpected niche.

Without the requirement for TCP acknowledgments, data encoded with Luby transform codes travels faster than ordinary packets; this advantage becomes significant over long distances. Thus, the technology has proved particularly useful to companies that frequently transmit extremely large data files over long distances, such as movie studios and oil exploration companies.

Indeed, Siebel, a large software developer, is using Digital Fountain technology for a collaborative software development project in which the data is constantly moved back and forth between Ireland and the U.S. Siebel's server paid for itself in about a week in gained productivity, Shokrollahi says.

"We don't believe this is the market that will take us public," Luby says, "but it's a big enough market to make us profitable." Eventually, however, he expects that Digital Fountain's technology might be used for distributing popular game software, distributing materials for distance learning, or even serving movies on demand.

For Luby, meanwhile, owning a company has meant a dramatic change from the low-key lifestyle of the academic. "In some ways, it's much more interesting and fulfilling than the research world because you have a real impact on things," he says. "The bad part is the stressed, crazy life."

"I'm glad I did it," he adds, "but I don't know if I would do it again."

Sara Robinson is a freelance writer based in Berkeley, California.