Research statement

Michael J. Freedman

My research interests span the areas of distributed systems, security, networking, and cryptography. I particularly enjoy devising technologies that make new functionality broadly available. My work generally tackles systems problems by coupling principled designs with real-world deployments.

A common thread in my research is the extension of systems designed for centralized or trusted entities into decentralized, untrusted, unreliable, or chaotic settings. These scenarios offer significant challenges, yet they are ones ideally suited for academic research: Such problems or architectures do not naturally arise from within industry, even though the techniques often may be applied back into managed environments, *e.g.*, to survive disasters or to operate safely under attack. More than that, open systems encourage further innovation.

I approach these problems through the innovative use of cryptography, algorithms, or abstractions. By leveraging the resulting properties, one can create self-organizing systems out of unreliable nodes, incentivize proper operation, curtail the impact of malicious behavior, or improve manageability to overcome system brittleness.

Such solutions still require solid engineering, always with the end-user in mind. By providing desired functionality, even research systems can attract users, gain traction, and then truly test the system's mettle. Deployed systems provide real data to direct future design decisions, and they can serve as platforms for otherwise intractable experiments. While much research relies solely on simulation and emulation, only at scale can we truly evaluate many systems—learning from their strengths, weaknesses, and emergent properties—and thus discover new research problems and directions.

Cooperative content distribution. My thesis research focuses on making content delivery more widely available by federating large numbers of untrusted or unreliable machines to share localized resources. Content distribution networks (CDNs) are not a new idea, but the architectures of commercial CDNs are tightly bound to centralized control, static deployments, and cost recovery.

My initial system, CoralCDN [1], explores how to build a self-organizing cooperative web CDN using unreliable hosts. Through its scalable distributed index, nodes can record and locate data without overloading any node, regardless of a file's popularity or system dynamics [1, 2]. Decentralized clustering algorithms enable nodes to find nearby data without querying more distant machines.

CoralCDN incorporates a number of engineering mechanisms for sharing resources fairly and preventing abuse—learned through deployment and community feedback—yet the system is inherently open. Simply modify a URL, and the requested content is automatically retrieved and cached by CoralCDN's proxies. As such, it has been widely adopted in often innovative ways: by servers to dynamically offload flash crowds, by browser extensions to recover from server failures, by podcasting and RSS software, and by daily links on Slashdot and other portals. CoralCDN currently handles about 25 million requests daily from over one million clients.

One challenge in designing CoralCDN was how to compel our unmodified clients to use nearby, unloaded proxies. While commercial systems also deploy *anycast* to select servers, their techniques need handle only a single deployment, often comprised of a mere handful of data centers. Ideally, one public infrastructure could provide anycast for many far-flung services, such that the more services that use it, the more accurate its server-selection results and the lower the bandwidth cost per service.

I built a subsequent system, OASIS [3], that does exactly this: OASIS currently provides anycast among thousands of servers from more than a dozen distributed systems, from both the academic and open-source communities. It flexibly supports a variety of interfaces—currently DNS, HTTP, and RPC—with which clients can discover good servers belonging to the requested system. OASIS can do so because it tackles several problems simultaneously: using nodes from participating services to perform network measurement, detecting and disambiguating erroneous results, representing locality stably across time and deployment changes, and scalably managing state information about many services.

This success at building content delivery from unreliable resources raised the question as to whether we could extend this approach to mutually distrustful clients. Shark [4] provides a distributed file system that improves scalability and performance through cooperative reads, using Coral's indexing layer to locate content. Still, Shark preserves traditional semantics and security: End-to-end cryptography ensures that clients need not trust one another.

We also considered security mechanisms for hosts using rateless erasure codes for cooperative large file distribution. Unfortunately, these codes cannot use traditional authenticators (*e.g.*, hash trees) that guarantee the integrity of individual blocks. Therefore, we devised a homomorphic hash function that can be used to verify downloaded blocks on-the-fly, thus preventing malicious participants from polluting the network with garbage [5]. Implementation aspects mattered in this seemingly-theoretical project. The batching of public-key operations was needed to achieve fast verification, while disk-read strategies led to encoding speeds that even exceeded those of hash trees for non-rateless codes. Finally, for preventing pollution in these non-rateless codes, we showed how simple implementation changes could replace others' heavyweight *black-box* mechanisms.

Recently, I have returned to the problem of moving CoralCDN from its current deployment on PlanetLab onto fully untrusted nodes, as CoralCDN's success has led to bandwidth usage that has long saturated PlanetLab's available capacity. As digital signatures can guarantee content integrity, the challenge is ensuring that sufficient capacity exists. Our latest design promotes resource sharing through incentive-compatible mechanisms: Contributing nodes receive better quality-of-service when the system is underprovisioned. The system applies market pricing techniques to efficiently use available bandwidth, but also incorporates network costs to "play friendly" with service providers. Malicious parties cannot cheat as lightweight cryptographic currency accurately tracks nodes' contributions.

While most of my work on cooperative content distribution has focused on leveraging unreliable or untrusted resources, I am not rigid in my approach. Indeed, some of these systems use logically-centralized components, such as the core OASIS infrastructure or, for each file collection in this last system, servers that manage file prices and currency exchange. Rather, I look where it is sensible or economical to leverage available resources—*e.g.*, local bandwidth for CDNs or measurement points for anycast—and architect systems accordingly. Indeed, these same cost arguments are behind industry's increased interest in such architectures, albeit without the same consideration for security.

Securing decentralized systems. When large decentralized systems lack the necessary security mechanisms, things eventually go awry. The Internet's inter-domain routing protocols (BGP) lack source authentication and thus routes have been hijacked, a weakness shared by DNS. Persistent email spam is frustrating, while false positives from spam filters have made email unreliable. Centralized solutions are not the only answer, however.

Tackling the spam false-positive problem, Re: [6] uses proximity in a social network as a basis for auto-whitelisting email. This approach appears promising given our analysis of large email corpora. And by incorporating our cryptographic protocols for private matching [7, 8], Re: ensures that two parties can maintain privacy without third-party intervention.

In a similar vein, websites want to securely identify their users, but ubiquitous client authentication does not exist. Thus, sites often use weaker identifiers such as IP addresses for access-control decisions, even though edge technologies (NATs, proxies, and DHCP) occlude a server's view of its clients. By instrumenting CoralCDN, we used active web content to measure and analyze the characteristics of over 7 million clients; our results help quantify when and how Internet services can use IP addresses and related information to identify clients [9]. (In fact, our techniques for real-time proxy detection and geolocation were acquired by a leading IP analytics company [10].) Here we see how a system, once widely used, can become a vehicle for otherwise infeasible research. Indeed, we are starting to investigate advertisement click fraud using this platform.

Enterprise networks similarly lack comprehensive security "from the ground up." Instead, a bewildering array of mechanisms (firewalls, NATs, and VLANs) have been retrofitted over the years, leading to brittle, inflexible networks. Begun as a clean-slate design [11], Ethane provides a backwards-compatible protection and management architecture for enterprise networks, where switches establish virtual circuits per flow, after using a domain controller to enforce security policies. Because Ethane simplifies so many network management tasks—testing new policies, deploying new appliances or topologies, performing forensics or fault diagnosis, establishing network isolation classes—its architecture empowers innovation and change within networks. I am further interested in extending such techniques to the wider area for managing autonomous systems.

Future work. Given the challenges of securing and managing networked systems, I have begun to think about new ways to simplify this task.

How can we determine when, where, and why performance or persistent faults in distributed systems occur? I intend to explore lightweight distributed tracing to track transactions across hosts and within processes. By tainting network communication and annotating code, we can generate system-wide "call graphs" during run-time. Of particular interest are identifying normal and anomalous system behavior, possibly through machine learning, and building feedback loops for automated reconfiguration. Other approaches to fault monitoring, detection, and diagnosis may be similarly promising. Of course, having deployed systems to test such tools is a critical advantage to experience the vagaries of failures in production environments. (In fact, others have used CoralCDN for exactly this [12].)

What new abstractions can provide better reliability in the face of failures? I am currently thinking about how to partition large systems into smaller groups, which can then apply heavyweight fault-tolerance or detection protocols [13]. (Such partitioning appears necessary for scalability.) While handling malicious parties in dynamic settings presents many difficult problems, the goal remains for better operation on faulty resources.

Finally, what privacy-preserving technologies can promote greater information sharing? Researchers, operators, and end-users can all benefit from greater access to data, whether inter-domain routing policies for traffic engineering, patient records for medical research, census and other polling data for the social sciences, or social information for cooperative filtering [6]. Unfortunately, privacy concerns often limit data availability, leading to suggestions such as private matching [7] for merging terrorist watch lists [14]. Yet current general-purpose cryptographic solutions are too inefficient for large datasets, while statistical methods are often not sound. I am interested in leveraging specific application contexts to build better protocols (as done in [8]), as well as exploring interface and architecture design for privacy-preserving systems.

While technology trends may incrementally improve system performance, new techniques are needed to enhance security, scalability, reliability, and manageability. I tackle these problems by applying methods from cryptography, distributed algorithms, game theory, and other principled sources. But real solutions require real testing: My research will embrace both strong design and engineering components, even as new problems arise over time. This unusual dual approach already has enabled my research systems to provide tens of millions of people with their Internet fix, often in surprising ways. Through such deployments we can discover new problems, encourage further innovation, and ultimately make new functionality broadly available.

References

[1] **M. Freedman**, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *Proc. Networked Systems Design and Implementation (NSDI)*, pages 239–252, Mar 2004.

[2] **M. Freedman** and D. Mazières. Sloppy hashing and self-organizing clusters. In *Proc. International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 45–55, Feb 2003.

[3] **M. Freedman**, K. Lakshminarayanan, and D. Mazières. OASIS: Anycast for any service. In *Proc. NSDI*, pages 129–142, May 2006.

[4] S. Annapureddy, **M. Freedman**, and D. Mazières. Shark: Scaling file servers via cooperative caching. In *Proc. NSDI*, pages 129–142, May 2005.

[5] M. Krohn, **M. Freedman**, and D. Mazières. On-the-fly verification of rateless erasure codes for efficient content distribution. In *Proc. IEEE Security and Privacy*, pages 226–240, May 2004.

[6] S. Garriss, M. Kaminsky, **M. Freedman**, B. Karp, D. Mazières, and H. Yu. Re: Reliable email. In *Proc. NSDI*, pages 297–310, May 2006.

[7] **M. Freedman**, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology — EUROCRYPT 2004*, pages 1–19, May 2004.

[8] **M. Freedman** and A. Nicolosi. Efficient private techniques for verifying social proximity. In *Proc. IPTPS*, Feb 2007.

[9] M. Casado and **M. Freedman**. Peering through the shroud: The effect of edge opacity on IP-based client identification. In *Proc. NSDI*, Apr 2007.

[10] Quova. http://www.quova.com/, 2006.

[11] M. Casado, T. Garfinkle, A. Akella, **M. Freedman**, D. Boneh, N. McKeown, and S. Shenker. SANE: A protection architecture for enterprise networks. In *Proc. USENIX Security Symposium*, pages 137–151, Aug 2006.

[12] P. Reynolds, J. Wiener, J. Mogul, M. Aguilera, and A. Vahdat. WAP5: Black-box performance debugging for wide-area systems. In *Proc. WWW*, May 2006.

[13] **M. Freedman**, I. Stoica, D. Mazières, and S. Shenker. Group therapy for systems: Using link-attestations to manage failures. In *Proc. IPTPS*, Feb 2006.

[14] J. Dempsey and P. Rosenzweig. Technologies that can protect privacy as information is shared to combat terrorism. Heritage Foundation Legal Memo #11, May 26 2004.