Keith Winstein — Research Statement

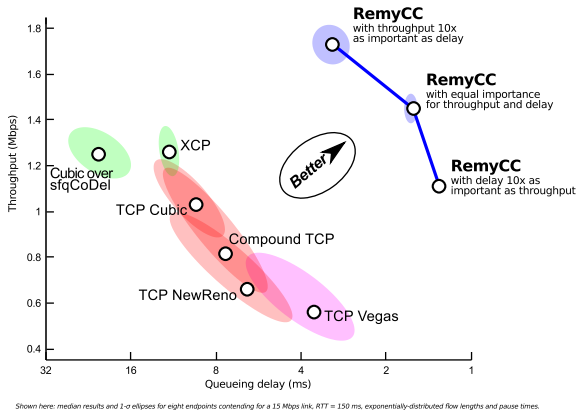
Over the last 30 years, few developments have altered life in the industrialized world as much as the Internet. My research applies tools from artificial intelligence and statistics to build networked systems that sustain this flood of innovation. My work expresses something of a common credo: (1) **explicitness in assumptions and objectives** frees systems to evolve, (2) we can **teach computers to design** systems from first principles and use the challenges we encounter to gain insight about the problem, and (3) devices are mobile and the network is unreliable, which means **applications benefit from richer information about the network** than conventional abstractions provide.

These themes have borne fruit in several areas. TCP ex Machina [1, 3] shows that a computer "designer" can generate competitive congestion-control schemes from first principles, helping us make explicit much of the hard-won intuition of the field. Sprout [4] is a protocol for interactive applications that forecasts network congestion in advance, reducing delay 7- to 9-fold compared with Skype, Hangouts, or Facetime. Mosh [2] is a popular remote-terminal application that uses client-side prediction to alleviate many of the frustrations of SSH (secure shell) over high-latency, mobile networks. Alfalfa, now in development, is a video-streaming system that applies decision theory to choose the best compressed video frames to download, eliminating the notion of video "streams."

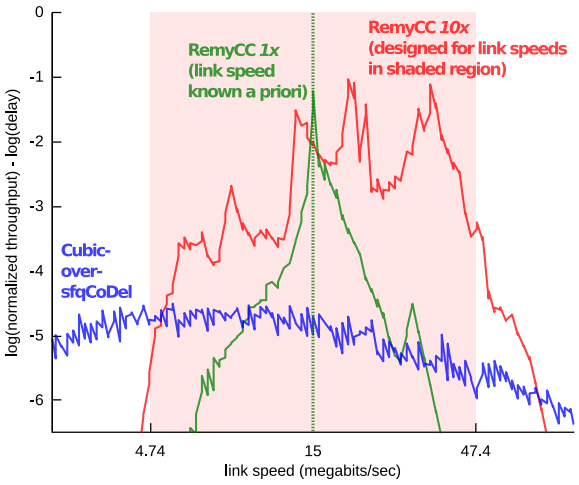
TCP ex Machina: The Internet is a decentralized network of diverse networks. In this architecture, transport protocols are the glue between an application's needs and the networks' varying abilities. The Transmission Control Protocol, or TCP, plays this role for the vast majority of Internet traffic, and TCP's implementations are among the most widely used computer programs in the world. As the Internet evolves to include ubiquitous wireless links and high-speed datacenters, and as users expect full-motion video and reliable service everywhere, the transport layer has begun showing its age. For any new network technology, it has become a de facto requirement that TCP perform well over it. But after three decades, these protocols' unwritten assumptions—about how typical networks behave and what users want—hold less and less well. This constrains the evolution of network technologies and applications.

My research on Inferential Transmission Control [1] and Remy [3] has found that by stating clearly the prior knowledge and goals of congestion control and searching systematically for the best solution, much of the past work in this area can be placed in a common framework and in some cases outperformed by protocols that a computer program generates in less than a day.

Remy-generated congestion control algorithms (RemyCCs, in blue) outperform existing human-generated systems, including schemes that benefit from intelligence inside the network (shown in green).



How helpful is prior knowledge about the network? Intuitively, we expect it to improve performance at the cost of reduced design range—but empirically the existence of a tradeoff is unclear.

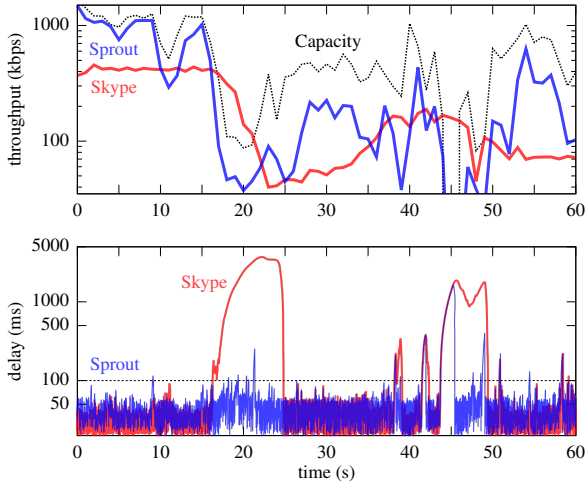


The benefits of "systematizing" the design process this way are manifold. By making the system a function of its assumptions about the network, we free network technologies to evolve and human designers to think about higher-level considerations. But the principal benefit of the Remy approach may be to help us materialize our intuitions—to use the computer as a tool to understand the job of systems design. What is TCP's real goal? Is there a price of staying "friendly" with today's TCP? Does prior knowledge about the network help, and by how much? How do we discuss the cost of designing with a simplified model in a complex world? These questions are as-yet unsolved by researchers in networking, artificial intelligence, and decentralized decision making.

Sprout: Cellular wireless networks have become a dominant mode of Internet access. The behavior of these networks differs markedly from the traditional wired Internet: their capacities vary rapidly in time, and they are semi-reliable and keep retransmitting packets until acknowledged. The combination of these two behaviors leads many applications to perform poorly—especially videoconferencing tools like Skype, Google+ Hangouts, and Apple's Facetime, which desire high throughput but don't want to create a backlog of packets that would add to video or audio delay.

My colleagues and I created Sprout, which models the variation in the network as Brownian motion and infers the current speed with uncertainty—then makes a stochastic forecast of the speed in the near future. Sprout provides this network forecast to the application, which can use it to pursue an explicit objective: send as much data as possible, but keep each packet's delay less than a tenth of a second with 95% probability. Although Sprout's model is simple, this approach produces 2-to-4 times the throughput, and 7-to-9 times less delay, than Skype, Hangouts, and Facetime.

Skype and Sprout on an LTE network trace. For Skype, overshoots in throughput lead to large standing queues. Sprout's forecasts let it anticipate slowdowns and respond cautiously so as not to build up a backlog of packets inside the network.



Sprout vs.	Avg. speedup	Delay reduction
Skype	2.2×	7.9×
Hangouts	4.4×	7.2×
Facetime	1.9×	8.7×
Compound	1.3×	4.8×
Vegas	1.1×	2.1×
LEDBAT	Same	2.8×
Cubic	0.91×	79×

Mosh: Mosh is a "gracefully mobile" replacement for SSH (secure shell) that allows roaming, supports intermittent connectivity, and provides an appropriate user interface for communication over an unreliable network. The main technical contributions are the use of single-packet roaming, which makes it easy to retain a connection when switching network addresses, and sliding prediction windows that let the Mosh client stay interactive even over networks that freeze tools such as SSH. Like Sprout, Mosh benefits from a richer interface with the network: a good predictive user interface needs to know which keystrokes have been acknowledged so far, as well as the current round-trip time. Mosh's user experience has been taught as an example of good mobile design in MIT's user interface class.

Mosh is widely deployed and has more than 500,000 users on GNU/Linux, Mac OS X, Microsoft Windows, BSD, Solaris, and Android, as well as a compatible client on iOS. The next version will allow users to opt in to submit measurements of their connections over time, creating a new data set that captures actual mobility and network quality across a diverse population of users and global network paths. The data will be useful to network researchers and protocol designers—including users of Remy—and will let us get at the question: as seen by its users, how is the Internet changing over time?

Alfalfa: My colleagues and I are working to apply an explicit decision-theoretic approach to the problem of online video streaming. By most measures, applications like YouTube, Netflix, and Hulu constitute the majority of Internet traffic. These systems play back videos without fully downloading them. Their behaviors represent assumptions about future network variability and the right tradeoff between the startup delay, the visual quality, the risk that video playback will stall in the middle, and the cost of data transmission.

Alfalfa makes those assumptions explicit by modeling the network's variability and the user's preference for fast startup, high quality, or smooth playback. Alfalfa augments the VP8 video compression standard to make it possible to switch quality levels and compression rates at every frame boundary, letting the player choose which coded frames to download to maximize the objective function. This requires optimizing a partially-observable Markov decision process (POMDP) at runtime. As with Remy, Sprout, and Mosh, the goal is to produce a principled streaming video system whose performance is a function of its well-stated assumptions and objectives.

Future plans: *Systems ex Machina.* My future work will use the Remy approach to probe fundamental questions in network science and systems design. How can we characterize the robustness of a computer-generated system that was designed for a particular model of the world and will necessarily be applied to a more-complex real world? Can we discover correspondences between superficially-dissimilar families of systems—for example, from the perspective of an endpoint, is robustness to variation in cross traffic the same thing as robustness to variation in link speed? Or can the collection of two-bottleneck topologies be modeled successfully as an uncertain one-bottleneck topology? Rather than optimizing merely for the best expected value of an objective, can we also teach a computer to design "fail-safes" that ensure sane behavior in edge cases? The notion of making systems-design problems precise and searching automatically for solutions is broadly applicable.

We will use Remy to develop and deploy new transport protocols across the Internet and inside data centers, and the challenges we encounter in real-world deployment will inform future design efforts. (In some sense, the more challenging, the more informative!) The data set from Mosh users about the performance of real-world networks will help to design the next generation of Internet transport protocols, letting us set assumptions and select parameters (like the size of the initial window) in principled, data-driven fashion.

App-centric networking. We have an opportunity to develop better protocols, motivated by application-layer objectives, that will be useful to practitioners and industry. Following up on Mosh, my collaborators in the open-source community and I are developing "Slosh" to bring roaming, tolerance for flaky networks, and multipath routing to all network applications, using Linux's lightweight containers. At MIT, we are developing "Mahimahi," a suite of tools to measure real-world Web performance reproducibly on emulated networks and understand how new transport-layer protocols (e.g. Sprout, a RemyCC, or Google's SPDY and QUIC) affect application-layer metrics like page load time.

Internet-native user interfaces. I will continue to develop "gracefully mobile" user interfaces that reflect the uncertainty of communicating over an unreliable network. Many of today's apps (including SSH, Gmail, Skype, and YouTube) simply freeze up or show stale data when network connectivity is lost. As the Internet expands globally and users expect good behavior from connections that cross oversubscribed wireless spectrum, there will be more opportunities to plumb richer network information to the application so it can respond to network variability and inform the user as appropriate. In a videoconference, no user should have to ask, "What was the last thing you heard?" or "Is the problem on your end or mine?" The program should make this obvious.

With Alfalfa, our belief is that objective-driven video streaming will produce dramatically better results than YouTube, Netflix, or Hulu—but to follow through, we'll need to come up with a user experience that elicits what the viewer cares about without overwhelming them with control. Are they willing to wait a bit longer for Blu-ray quality or an assurance against stalls, or do they just want a quick preview?

Decentralized video streaming. If Alfalfa is successful as a high-performance replacement for today's video streaming technology, it has the potential to be deployed broadly as a *decentralized* video streaming service, i.e. as YouTube without a central YouTube. This would require sharing encoding and hosting responsibilities equitably across network participants, so that more-popular videos receive more CPU cycles and consequently better playback and quicker startup for future viewers. To do this successfully with YouTube-like performance will require solving fundamental problems in distributed systems and mechanism design.

Reproducibility. My research publications have come with software that the reader can run (Mosh, Sprout, Remy) and replication packages that reproduce the major results and figures of the paper. I think this is a healthy practice to continue. The logical next step is to continue the conversation beyond publication by hosting a site where anyone can propose a new system, evaluate it on the same metrics, and see everybody's results plotted on the same axes.

[1] Keith Winstein and Hari Balakrishnan. End-to-End Transmission Control by Modeling Uncertainty about the Network State. In *ACM HotNets-X*, 2011.

[2] Keith Winstein and Hari Balakrishnan. Mosh: An Interactive Remote Shell for Mobile Clients. In *USENIX Annual Technical Conference*, 2012.

[3] Keith Winstein and Hari Balakrishnan. TCP ex Machina: Computer-Generated Congestion Control. In *ACM SIGCOMM*, 2013.

[4] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *USENIX NSDI*, 2013.