An Experimental Study of the Learnability of Congestion Control

Anirudh Sivaraman, Keith Winstein, Pratiksha Thaker, and Hari Balakrishnan
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, Mass.
{anirudh, keithw, prthaker, hari}@mit.edu

ABSTRACT

When designing a distributed network protocol, typically it is infeasible to fully define the target network where the protocol is intended to be used. It is therefore natural to ask: How faithfully do protocol designers really need to understand the networks they design for? What are the important signals that endpoints should listen to? How can researchers gain confidence that systems that work well on well-characterized test networks during development will also perform adequately on real networks that are inevitably more complex, or future networks yet to be developed? Is there a tradeoff between the performance of a protocol and the breadth of its intended operating range of networks? What is the cost of playing fairly with cross-traffic that is governed by another protocol?

We examine these questions quantitatively in the context of congestion control, by using an automated protocol-design tool to approximate the best possible congestion-control scheme given imperfect prior knowledge about the network. We found only weak evidence of a tradeoff between operating range in link speeds and performance, even when the operating range was extended to cover a thousand-fold range of link speeds. We found that it may be acceptable to simplify some characteristics of the network—such as its topology—when modeling for design purposes. Some other features, such as the degree of multiplexing and the aggressiveness of contending endpoints, are important to capture in a model.

CATEGORIES AND SUBJECT DESCRIPTORS

C.2.2 [**Computer-Communication Networks**]: Network Protocols

KEYWORDS

Protocol; Machine Learning; Congestion Control; Learnability; Measurement; Simulation

1. INTRODUCTION

Over the last 30 years, Internet congestion control has seen considerable research interest. Starting with seminal work in the 1980s [24, 15, 7], the Transmission Control Protocol (TCP) has adopted a series of end-to-end algorithms to share network resources among contending endpoints [14, 6, 12, 27]. Another line of work has explored the use of in-network algorithms running at bottleneck routers to help perform this function more efficiently [11, 10, 9, 18, 23, 21, 16].

As the Internet grows and evolves, it appears likely that new network protocols will continue to be developed to accommodate new subnetwork behaviors and shifting application workloads and goals. Some of these may be intended for specialized environments—e.g., inside a centrally-managed datacenter—while some will be for broad use across the wide-area Internet, or over cellular network paths.

In practice, however, it is challenging to guarantee that a distributed system's performance on well-characterized test networks will extend to real networks, which inevitably differ from those envisioned in development and will continue to evolve over time. This uncertain generalizability presents an obstacle to any new protocol's deployment.

In this paper, we formalize the design process for generating end-to-end congestion-control protocols to quantify: *how easy is it to "learn" a network protocol to achieve desired goals, given a necessarily imperfect model of the networks where it will ultimately be deployed?*

Under this umbrella, we examine a series of questions about what knowledge about the network is important when designing a congestion-control protocol and what simplifications are acceptable:

1. **Knowledge of network parameters.** Is there a tradeoff between the performance of a protocol and the breadth of its intended operating range of network parameters [30]? Will a "one size fits all" protocol designed for networks spanning a broad range of link speeds (§4.1), degrees of multiplexing (§4.2), or propagation delays (§4.3) necessarily perform worse than one targeted at a narrower subset of networks whose parameters can be more precisely defined in advance?

2. **Structural knowledge.** How faithfully do protocol designers need to understand the topology of the network? What are the consequences of designing protocols for a simplified network path with a single bottleneck, then running them on a network with two bottlenecks (§4.4)?

3. **Knowledge about other endpoints.** In many settings, a newly-designed protocol will need to coexist with traffic from other protocols. What are the consequences of designing a protocol with the knowledge that some endpoints may be running incumbent protocols whose traffic needs to be treated fairly (e.g. the new protocol needs to divide a contended link evenly with TCP), versus a clean-slate design (§4.5)? Similarly, what are the costs and benefits of designing protocols that play nicely with traffic optimized for other

objectives—e.g., a protocol that aims for bulk throughput but wants to accommodate the needs of cross-traffic that may prioritize low delay (§4.6)?

4. **Knowledge of network signals.** What kinds of congestion signals are important for an endpoint to listen to? How much information can be extracted from different kinds of feedback, and how valuable are different congestion signals toward the protocol's ultimate goals? (§3.4)

1.1 Tractable attempts at optimal (Tao) protocols

Each of the above areas of inquiry is about the effect of a protocol designer's imperfect understanding of the future network that a decentralized congestion-control protocol will ultimately be run over.

In principle, we would quantify such an effect by evaluating, on the actual network, the performance of the "best possible" congestion-control protocol designed for the imperfect network *model*, and comparing that with the best-possible protocol for the actual network itself.

In practice, however, we know of no tractable way to calculate the optimal congestion-control protocol for a given network.[1] Instead, throughout this study we use the Remy automatic protocol-design tool [29], a heuristic search procedure that generates congestion-control protocols, as a proxy for the optimal solution.

We refer to these automatically generated congestion-control protocols as "tractable attempts at optimal" (Tao) end-to-end congestion-control protocols for a given network. Tao protocols represent a practically realizable tilt at developing an optimal protocol for an imperfect model of a real network.

Constructing a Tao for a complex network model requires searching a huge space, an intensive task even using Remy's search optimizations. In most cases, the protocol stopped improving within a CPU-year of processing time (five days on our 80-core machine), though there were a few cases where improvements continued to occur.

We emphasize that there can be no assurance that the Tao actually comes close to the optimal congestion-control protocol, except to the extent that it approaches upper bounds on performance, such as the ideal fair allocation of network resources. To characterize how close the Tao protocols come to the bound, we formalize the notion of a hypothetical "omniscient" protocol. This is a centralized protocol that knows the topology of the network, the link speeds, the locations of senders and receivers, and the times at which they turn on or off. Each time a sender turns on or off, the omniscient protocol computes the proportionally fair throughput allocation [17] for all senders that are on. Each sender then transmits at its proportionally fair throughput allocation, and no flow builds up a standing queue at any bottleneck. For a particular node, the long-term average throughput of the omniscient protocol is the expected value of its throughput allocation, with no queueing delay.

As a calibration experiment to validate the Tao approach, we design a Tao protocol with parameters shown in Table 1 (where the buffer size of 5 BDP refers to 5 times the bandwidth-delay product: the network link speed times the minimum round trip time (RTT)). We summarize the performance of a protocol using a throughput-delay graph as shown in Figure 1. For each protocol, we plot the median throughput and delay (small white circle) and an ellipse representing one standard deviation of variation. The Tao protocols are considerably better than two human-designed protocols: Cubic [12] (the default end-to-end congestion-control algorithm on Linux) and Cubic over sfqCoDel (Cubic coupled with sfqCoDel [2], a recent proposal for queue management and scheduling that runs on the gateway nodes) on both throughput and delay. Furthermore, they come within 5% of the omniscient protocol on throughput and 10% on delay.

[1] The problem can be formulated as a search procedure for an optimal policy for a decentralized partially observable Markov decision process or Dec-POMDP [29]. However, the search for an optimal policy under a general Dec-POMDP is known to be NEXP-complete [4], and no faster search procedure for the particular problem of network-protocol design has been found.

Link speed	32 Mbits/sec
Minimum RTT	150 ms
Topology	Dumbbell
Senders	2
Workload	1 sec ON/OFF times
Buffer size	5 BDP
Objective	$\sum log(tpt) - log(delay)$

Table 1: Network parameters for the calibration experiment.

The calibration experiment does not prove that the Tao protocols will continue to approach omniscient performance when there is uncertainty about the network scenario at runtime, or when the network scenarios are more complex. It also does not demonstrate the relationship between the Tao protocols and the true "optimal" solution, which (unlike the omniscient protocol) is one that will be realizable in an end-to-end algorithm when endpoints have only partial information about the true network state.

However, the results do give confidence that the Tao protocols can be used as tractable proxies for an optimal protocol. In this study, we compare human-designed protocols to various Tao protocols and the omniscient protocol, generally finding that Tao protocols can approach the omniscient protocol and can outperform the existing human-designed protocols that have been designed to date (Figures 2, 3, 4, and 6).

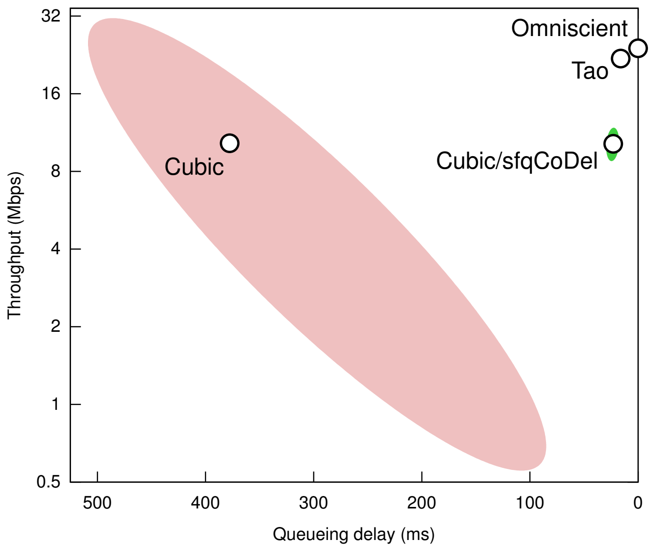


Figure 1: In the calibration experiment, the Tao protocol approached the performance of the omniscient protocol.

1.2 Summary of results

Here are the principal findings of this study:

Modeling a two-bottleneck network as a single bottleneck hurt performance only mildly. On the two-hop network illustrated by Figure 5, on average we find that a protocol designed for a simplified, single-bottleneck model of the network underperformed by **only 17%** (on throughput) a protocol that was designed with full knowledge of the network's two-bottleneck structure (Figure 6). Furthermore, the simplified protocol also outperformed TCP Cubic by a factor of 7.2× on average throughput and outperformed Cubic-over-sfqCoDel by 2.75× on average throughput.

Thus, in this example, full knowledge of the network topology during the design process was not crucial.

Weak evidence of a tradeoff between link-speed operating range and performance. We created a Tao protocol designed for a range of networks whose link speeds spanned a thousand-fold range between 1 Mbps and 1000 Mbps, as well as three other protocols that were more narrowly-targeted at hundred-fold, ten-fold, and two-fold ranges of link speed (Figure 2).

The "thousand-fold" Tao achieved close to the peak performance of the "two-fold" Tao. Between link speeds of 22–44 Mbps, the "thousand-fold" Tao achieved **within 3% of the throughput** of the "two-fold" protocol that was designed for this exact range. However, the queueing delay of the "thousand-fold" protocol was **46% higher**, suggesting some benefit from more focused operating conditions. It also takes a lot longer to compute (offline) a "thousand-fold" Tao compared to a two-fold Tao; in run-time operation, though, the computational cost of the two algorithms is similar.

Over the full range of 1 Mbps to 1000 Mbps, the "thousand-fold" Tao protocol matched or outperformed TCP Cubic and Cubic-over-sfqCoDel over the entire range (Figure 2).

The results suggest that highly optimized protocols may be able to perform adequately over a broad range of actual networks. Broad operating range had only a weak effect on peak performance, suggesting that "one size fits all" congestion-control protocols that perform considerably better than TCP—as well as TCP plus in-network assistance, in the case of sfqCoDel—may be feasible.

Performance at high degrees of multiplexing may be in opposition with performance when few flows share a bottleneck. We created five Tao protocols for a range of networks with varying degrees of multiplexing: 1–2 senders, 1–10, 1–20, 1–50, and 1–100 (Figure 4.2 and Table 3a).

We found that a Tao protocol designed to handle between 1 and 100 senders came close to the performance achieved by an omniscient protocol over most of that range. However, this came at the cost of diminished throughput at lower degrees of multiplexing. Conversely, a protocol trained to handle between 1 and 2 senders suffered large queuing delays (when the link never dropped packets) or repeated packet drops (on a link with finite buffering) when run on a network with 100 senders.

These results suggest that—unlike with link speed—prior knowledge of the expected degree of multiplexing over bottleneck links may be beneficial when designing a congestion-control protocol.

TCP-awareness hurt performance when TCP cross-traffic was absent, but helped dramatically when present. We measured the costs and benefits of "TCP-awareness"—designing a protocol with the explicit knowledge that it may be competing against other endpoints running TCP, compared with a "TCP-naive" protocol for a network where the other endpoints only run the same TCP-naive protocol.

When contending only with other endpoints of the same kind, the "TCP-naive" protocol achieved **55% less queueing delay** than the TCP-aware protocol. In other words, "TCP-awareness" has a cost measured in lost performance when TCP cross-traffic is not present (Figure 7).

But when contending against TCP, the "TCP-naive" protocol was squeezed out by the more aggressive cross-traffic presented by TCP. By contrast, the "TCP-aware" protocol achieved **36% more throughput** and **37% less queueing delay** than the "TCP-naive" protocol, and claimed its fair share of the link capacity from TCP.

Instructions to reproduce the results in this paper, along with the congestion-control protocols produced by Remy in the process are available at http://web.mit.edu/remy/learnability.

2. RELATED WORK

This section discusses closely related work on congestion control and explains how our work has an analogy with theoretical notions of learnability.

2.1 Congestion control

Congestion control over packet-switched networks has been a productive area of research since the seminal work of the 1980s, including Ramakrishnan and Jain's DECBit scheme [24] and Jacobson's TCP Tahoe and Reno algorithms [15]. End-to-end algorithms typically compute a congestion window or a paced transmission rate using the stream of acknowledgments (ACKs) arriving from the receiver. In response to congestion, inferred from packet loss or, in some cases, rising delays, the sender reduces its window or rate; conversely, when no congestion is perceived, the sender increases its window or rate.

In this paper, we use the Remy [29] protocol-design tool to generate end-to-end Tao congestion-control schemes from first principles. The Remy work showed that such an approach can produce schemes whose performance is competitive with or outperforms human-generated schemes, including most varieties of TCP congestion control, on intended target networks.

By contrast, this paper uses the Remy program as a tool for understanding the nature of the problem of protocol design without being able to fully define the intended target networks. We use the program's output as a proxy for the "best possible" Tao congestion-control protocol intended for a particular imperfect network model, and then ask how that protocol performs on a different set of networks that varies from the model in some respect (topology, link speed, behavior of contending endpoints, etc.).

For reference, we also compare with existing congestion-control protocols in wide use, including Linux's TCP Cubic [12], and the less-aggressive NewReno algorithm [14]. End-to-end congestion control may be assisted with explicit router participation; we also measure Cubic in conjunction with sfqCoDel [2]. sfqCoDel runs at the bottleneck gateways and uses the CoDel [21] queue management algorithm along with the stochastic fair queueing [20] scheduling algorithm.

2.2 Learnability

TCP congestion control was not designed with an explicit objective function in mind. Kelly et al. present an interpretation of TCP congestion-control variants in terms of the implicit goals they attempt to optimize [17]. This line of work is known as Network Utility Maximization (NUM); more recent work has modeled stochastic NUM problems [31], where a stochastic process dictates how flows enter and leave the network over time.

We extend this problem by examining the difficulty of designing a network protocol given an *imperfect* model of the network where it will be deployed, in order to understand the inherent difficulties of the problem of congestion control.

Formally speaking, designing such a protocol is a problem in sequential decision-making under uncertainty and can be modeled as a decentralized partially observable Markov decision process [4] (Dec-POMDP). In that context, the purpose of this paper is to ask: how well can a protocol designer "learn" the optimal policy (protocol) for one Dec-POMDP on a given set of networks and successfully apply the learned policy to a different set of networks?

In doing so, we draw an explicit analogy to the concept of "learnability" employed in machine learning [28, 25]. A canonical machine-learning task attempts to design a classifier for a large population of data points, supplied with only a smaller (and possibly skewed) "training set" meant to teach the classifier about the full population of data. Subsequently, the performance of the resulting classifier is evaluated in how well it correctly classifies points in a "test set", generally drawn from the actual population. Learnability theory measures the difficulty of inferring an accurate classifier for the test set, given a training set.

Just as a classifier-design procedure may minimize the error rate or maximize the width of the margin [5] over the training set as a proxy for maximizing predictive performance on unseen inputs, the Remy tool uses an objective function in terms of throughput and delay, averaged over the design model, as a proxy for performance on as-yet-unseen networks.

In our work, we envision a protocol designer working to generate a congestion-control protocol for a large set of real networks (e.g., the Internet), supplied with only an imperfect model of the range of variation and behavior of those networks. The imperfect model is the "training scenarios"—a model of the target networks used for design purposes. The "testing scenarios" are drawn from the population of actual target networks. In contrast with theoretical notions of learnability that rigorously demonstrate the learnability of entire families of functions [28], this study assesses learnability experimentally: we measure the difficulty (in terms of lost performance) of designing an adequate protocol for a network *model*, and then deploying it on target networks that cannot be perfectly envisioned at the time of design.

3. EXPERIMENTAL SETUP

We describe our experimental procedure below. First, we specify a set of *training scenarios*: a set of network configurations (§3.1) that express the designer's imperfect model of the network. Next, we specify an *objective function* (§3.2). The protocol-design process (§3.3) synthesizes a congestion-control protocol that maximizes the value of this function, averaged over the set of training scenarios. Finally (§3.6), we specify the *testing scenarios* of network configurations, which may be similar or dissimilar to the training scenarios.

We evaluate the synthesized congestion-control protocol on the testing scenarios to assess the questions of this study—how easy is it to "learn" a network protocol to achieve desired goals, given an imperfect model of the networks where it will ultimately be deployed?

3.1 Training scenarios

The training scenarios specify the set of network configurations that the protocol-design process is given access to. Formally, a network configuration specifies:

1. The topology: the link speed and propagation delay of each link in the network along with a graph representing the interconnections between nodes.

2. The locations of senders and receivers within the topology, and the paths connecting the senders to the receivers.

3. A model of the workload generated by the application running at each endpoint. We use an on/off model for the workload, where a sender turns "on" for a certain duration drawn from an exponential distribution, then turns "off" for an amount of time drawn from another exponential distribution before turning on again.

4. The buffer size and queueing discipline at each gateway. For all training scenarios in this paper, we model a FIFO queue. For testing scenarios, we model a FIFO queue except in the case of Cubic-over-sfqCoDel, which runs sfqCoDel at the gateway nodes.

3.2 Objective function

The objective function expresses the protocol designer's figure of merit for the goodness of a congestion-control protocol. Many such metrics have been proposed, including alpha-fair throughput [26], flow completion time [8], throughput-over-delay [22], or measures based on a subjective opinion score [13].

In this study, we specifically considered objective functions of the form:

$$\log(\text{throughput}) - \delta \log(\text{delay}) \qquad (1)$$

Here, "throughput" is the average information transmission rate of a sender-receiver pair, defined as the total number of bytes successfully delivered divided by the total time the sender was "on" and hence had offered load. The "delay" is the average per-packet delay of packets in the connection, including propagation delay and queueing delay. The δ factor expresses a relative preference between high throughput and low delay.

The protocol-design process works to maximize the *sum* of the objective function across all connections. The log in the objective function expresses a preference for "proportionally fair" resource allocation [17]—for example, it is worthwhile to cut one connection's throughput in half, as long as this allows another connection's throughput to be more-than-doubled.

3.3 Protocol-design process

We outline the Remy protocol-design tool briefly here, following the treatment of [29]. Remy models a congestion-control protocol as a set of match-action rules, mapping between the state maintained by the sender and an action to be executed. The "state" tracks a small set of congestion signals, updated on every acknowledgment from the receiver. The "action" specifies changes in the behavior of the congestion-control protocol.

To simplify learning, Remy assumes a piecewise-constant mapping, and searches for the mapping that maximizes the average value of the objective function across the training scenarios. The mapping is initialized to prescribe a default action for all memory values. Remy then simulates the protocol on the training scenarios and uses the simulation results—and the resulting value of the objective function—to gradually refine the mapping.

For the experiments in this paper, the sender tracks four congestion signals:

1. `rec_ewma`: An exponentially-weighted moving average, or EWMA, of the interarrival times between acks with a weight of 1/8 for new samples.

2. `slow_rec_ewma`: The same as `rec_ewma`, but with a weight of 1/256 for new samples, producing an average taken over a longer history.

3. `send_ewma`: A moving average of the intersend time between sender timestamps echoed in the received ACKs, with a weight of 1/8 for new samples.

4. `rtt_ratio`: The ratio of the most recent round-trip-time measurement and the minimum RTT seen so far.

3.4 Value of the congestion signals

We performed a measurement study to evaluate the value of each of these four signals on the ultimate performance of a congestion-control protocol. We selectively "knocked out" each signal in turn and designed a new congestion-control protocol from scratch (missing that signal), in order to observe the effect of losing the signal on the protocol's ultimate behavior.

In our measurements, we found that each of these congestion signals independently brought value to a congestion-control protocol. No three-signal subset was as strong as using all four signals. The most valuable signal—by which we mean the signal whose removal caused the greatest harm to the ultimate performance—was the `rec_ewma`. This suggests that these protocols may gain considerable value from understanding the short-term packet-arrival dynamics at the receiver.

3.5 The congestion response

The action uses a window-based congestion-control protocol that caps the number of packets in flight, with pacing to regulate the rate at which an end host meters packets into the network. The action for any value of the memory is a triplet that specifies:

1. A multiplier m to the current value of the congestion window.

2. An increment b to the current value of the congestion window.

3. A lower bound τ on the pacing interval between outgoing packet transmissions.

To emphasize that the resulting protocol is a brute-force approximation of the best algorithm for a given set of training scenarios and objective function, we refer to such protocols as "tractable attempts at optimal" congestion control, or Tao protocols.

3.6 Evaluation procedure

To measure the difficulty of learning a congestion-control protocol with an imperfect model of the eventual network, we choose a testing scenario of network configurations and evaluate the Tao protocol on it. All evaluations are performed in the ns-2 simulator. Using a different simulator for training and testing helps build confidence that the congestion-control protocols learned are robust to quirks in a simulator implementation.

We compare the performance of Tao protocols optimized for "accurate" models of the network against Tao protocols optimized for various kinds of imperfect models, in order to measure how faithfully protocol designers need to understand the network they are designing for. In the interest of reproducibility, for each experiment that we run, we tabulate the training and testing scenarios (for e.g. Tables 3a and 3b).

For reference, we also compare the Tao protocols with two common schemes in wide use today:

1. TCP Cubic [12], the default congestion-control protocol in Linux

2. Cubic over sfqCoDel [2], an active-queue-management and scheduling algorithm that runs on bottleneck routers and assists endpoints in achieving a fairer and more efficient use of network resources

3.7 Caveats and non-goals

We view this work as a first step towards answering questions under the broad umbrella of the "learnability" of congestion control. The simulation experiments are deliberately simplistic, favoring scenarios that explore foundational questions over simulations that are representative of deployed networks. We use Remy-generated protocols as a proxy for the optimal solutions, which means that the results may change when better protocol-design tools are developed, or when these protocols are tested on real physical networks outside of simulation. It is not our goal here to reverse-engineer the per-node, microscopic behavior of Tao protocols—we are interested in using measurements of their varying performance to characterize the learnability of the protocol-design problem itself.

4. INVESTIGATING THE LEARNABILITY OF CONGESTION CONTROL

4.1 Knowledge of link speed

We evaluated the difficulty of designing a congestion-control protocol, subject to imperfect knowledge of the parameters of the network.

Some congestion-control protocols have been designed for specific kinds of networks [3, 19] or require explicit knowledge of the link speed *a priori* [16]. Others are intended as a "one size fits all," including most variants of TCP.

We set out to answer a question posed in [30]: are "one size fits all" protocols inherently at a disadvantage, because of a tradeoff between the "operating range" of a protocol and its performance?

To quantify this, we designed four Tao protocols for training scenarios encompassing a thousand-fold variation in link speeds, a hundred-fold variation, a ten-fold variation, and a two-fold variation. Each range was centered on the geometric mean of 1 and 1000 Mbps (32 Mbps), and each set of training scenarios sampled 100 link speeds logarithmically from the range. The training and testing scenarios are shown in Table 2.

Tao	Link speeds	RTT	Number of senders
1000x	1–1000 Mbps	150 ms	2
100x	3.2–320 Mbps	150 ms	2
10x	10–100 Mbps	150 ms	2
2x	22–44 Mbps	150 ms	2

(a) Tao protocols designed for breadth in link speed

Link speeds	RTT	Number of senders
1–1000 Mbps	150 ms	2

(b) Testing scenarios to explore breadth in link speed

Table 2: Scenarios for "knowledge of link speed" experiment, showing the effect of varying the intended link-speed operating range. Each Tao was designed for a network with a single bottleneck, and each sender had a mean "on" and "off" time of 1 s.

We tested these schemes in ns-2 by sweeping the link speed between 1 and 1000 Mbps, keeping the other details of the simulated network identical to the training scenario. The results are shown Figure 2.

We find evidence of a weak tradeoff between operating range and performance—optimizing for a smaller range did help modestly over that range. However, the improvements in objective from narrowing down the operating range were modest, and each Tao

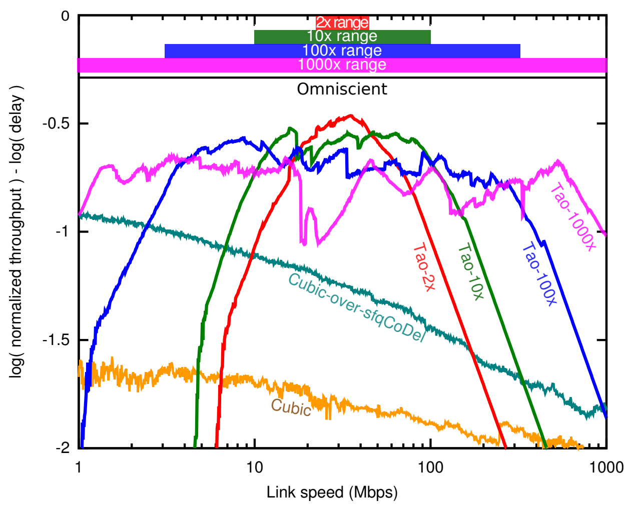


Figure 2: Evidence of a weak tradeoff between operating range in link speed of a congestion-control protocol and performance. The Tao protocols designed with more specific network models (Tao-2x and Tao-10x) performed modestly better—within their design ranges—than protocols designed for a broader range of networks (Tao-100x and Tao-1000x), at a cost of deterioration when the actual network did not fall within the training scenarios. The four Tao protocols outperformed Cubic and Cubic-over-sfqCoDel over their design ranges.

(including the Tao-1000x) outperformed existing algorithms over its full design range.

4.2 Knowledge of the degree of multiplexing

Congestion-control protocols have to cope with uncertainty in the degree of multiplexing from one use to the next even when running on a network with a fixed and known topology. The number of users on the network could change as could the number of flows generated by these users. Here, we evaluate whether it is possible to learn a protocol that is robust to such uncertainty.

Specifically, we designed five Tao protocols, each for a fixed dumbbell topology, but with increasing degrees of multiplexing as described in Table 3a. We then test all five protocols on the testing scenarios in Table 3b. Figure 3 shows the results for two cases: one with a buffer size of five times the bandwidth-delay product and an extreme case where the link doesn't drop any packet. In both cases, we see that it is possible to train a protocol to perform well across a wide range of multiplexing, but at the cost of diminished performance at very low degrees of multiplexing. On the flip side, designing for a much smaller range in the degree of multiplexing degrades performance dramatically outside that range.

Put differently, there is a tradeoff between good performance at high and low degrees of multiplexing. A high degree of multiplexing requires a protocol to be conservative in grabbing spare bandwidth and good performance at low degrees of multiplexing needs a protocol to quickly grab spare bandwidth. As a result, protocols trained for low degrees of multiplexing are too aggressive at higher degrees, causing either large queuing delays, when the buffer never drops a packet, or diminished throughput, when the buffer drops packets and causes more retransmissions than transmissions[2]. In either case, the result is a degradation in the objective function when such low-degree protocols are tested outside their training range. Conversely, a protocol trained for a wide range in the degree of multiplexing is unduly conservative at lower degrees of multiplexing leading to lost throughput and a degradation in the objective function.

[2]This is similar to the congestion collapse that triggered the development of TCP congestion control [15].

Tao	Link speeds	On avg.	Off avg.	min-RTT	# senders
Tao-1–2	15 Mbps	1 sec	1 sec	150 ms	2
Tao-1–10	15 Mbps	1 sec	1 sec	150 ms	10
Tao-1–20	15 Mbps	1 sec	1 sec	150 ms	20
Tao-1–50	15 Mbps	1 sec	1 sec	150 ms	50
Tao-1–100	15 Mbps	1 sec	1 sec	150 ms	100

(a) Tao protocols designed for breadth in multiplexing

Link speeds	On avg.	Off avg.	min-RTT	# senders	Buffer
15 Mbps	1 sec	1 sec	150 ms	1–100	5 BDP, no drop

(b) Testing scenarios to explore breadth in multiplexing

Table 3: Scenarios for "knowledge of the degree of multiplexing" experiment

4.3 Knowledge of propagation delays

Today, some specialized congestion-control protocols are tailored for networks with either large propagation delays (e.g., the Aspera [1] file transfer protocol over long-distance links) or short delays (such as DCTCP [3] within the data center). Here, we ask whether it is possible to design a single congestion-control protocol with good performance without making strong assumptions about the propagation delay *a priori*.

We designed four Tao protocols for varying ranges of the minimum round trip time of the network as shown in Table 4a and tested it on the scenarios shown in Table 4b. Figure 4 shows that training for exactly one minimum RTT (150 ms) results in a protocol with good performance over the 50–250 ms range, but its performance degrades drastically below 50 ms.

Tao	Link speeds	On avg.	Off avg.	min-RTT	# senders
rtt-150	33 Mbps	1 sec	1 sec	150 ms	2
rtt-145–155	33 Mbps	1 sec	1 sec	145–155 ms	2
rtt-140–160	33 Mbps	1 sec	1 sec	140–160 ms	2
rtt-50–250	33 Mbps	1 sec	1 sec	50–250 ms	2

(a) Tao protocols designed for breadth in propagation delay.

Link speeds	On avg.	Off avg.	min-RTT	# senders
33 Mbps	1 sec	1 sec	1, 2, 3 . . . 300 ms	2

(b) Testing scenarios to explore breadth in propagation delay.

Table 4: Scenarios for "knowledge of propagation delay" experiment

On the other hand, adding a little diversity to the training — by looking at minimum RTTs from 145 to 155 ms instead — results in performance over the 1–300 ms range that is commensurate with a protocol trained for the much broader range of 50 to 250 ms.

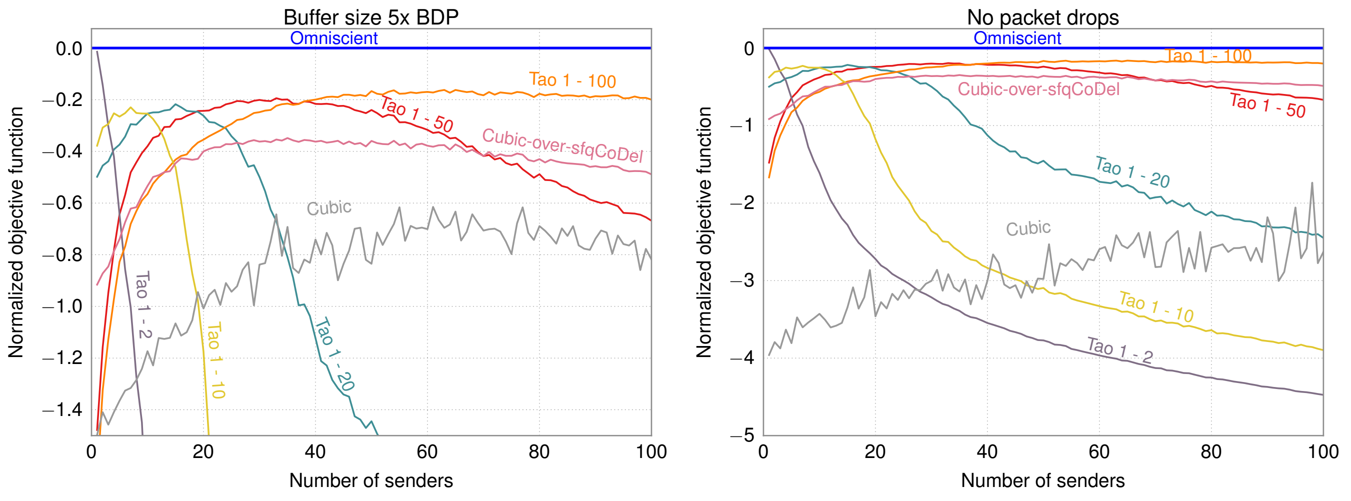


Figure 3: Tao protocols perform well across a wide range of multiplexing, but at the cost of diminished performance when there are very few senders. However, training to accommodate lower degrees of multiplexing degrades performance at higher degrees of multiplexing.

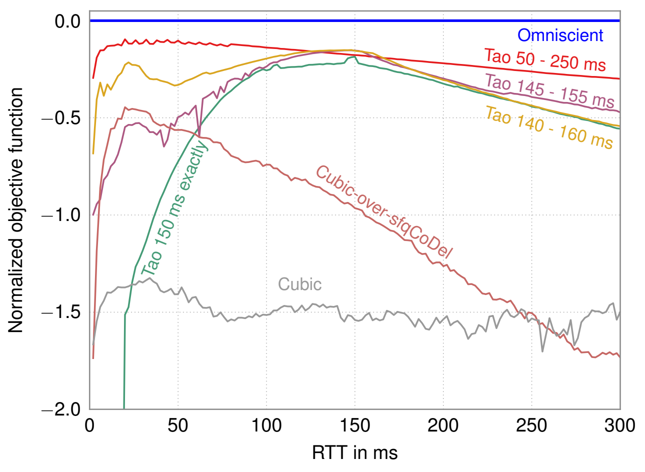


Figure 4: Training for a little diversity in propagation delays results in good performance over a much wider range of propagation delays.

These results suggest that prior knowledge of the propagation delay of a target network may not be particularly necessary or valuable to the protocol designer.

4.4 Structural knowledge

We evaluated the difficulty of designing a congestion-control protocol, subject to imperfect knowledge of the network's structure or topology.

It is an understatement to say that the Internet is a vast network whose full structure is known to nobody and which no model can accurately capture. Nonetheless, researchers regularly develop new distributed protocols for the Internet, which are deployed based on tests in example network paths that imperfectly capture the Internet's true complexity.

In practice, protocol designers expect that they can reason about the performance of a distributed network algorithm by modeling the network as something simpler than it is. We worked to capture that intuition rigorously by studying quantitatively how difficult it is to learn a congestion-control protocol for a more-complicated network, given a simplified model of that network's structure.

In ns-2, we simulated a network with two bottlenecks in a "parking-lot" topology, shown in Figure 5. Flow 1 crosses both links and encounters both bottlenecks. It contends with Flow 2 for access to the bottleneck queue at node A, and contends with Flow 3 for access to the bottleneck queue at node B.

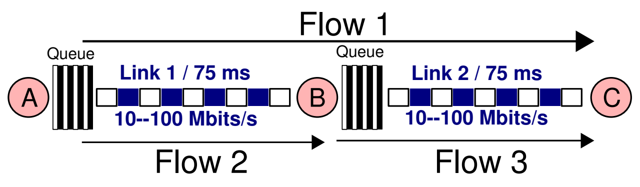


Figure 5: Parking-lot topology used to measure the consequences of imperfect knowledge of the network's structure.

Tao	Links modeled	Num. senders
one-bottleneck	one, 150 ms delay	2
full two-bottleneck	two, 75 ms delay each	3

Table 5: Training scenarios used to measure the consequences of imperfect knowledge of the network structure. Both protocols were designed for link speeds distributed log-uniformly between 10 and 100 Mbps, and for flows with mean "on" and "off" time of 1 second.

The results for Flow 1 (the flow that crosses both links) are shown in Figure 6.[3] The experiment sweeps the rate of each of the two links between 10 and 100 Mbps, and the shaded area in the figure shows the full locus of throughputs seen by Flow 1. For each pair of rates (for link 1 and link 2), we also calculate the ideal proportionally fair throughput allocation, and plot the locus of these allocations as the throughput achievable by the omniscient protocol.

The results suggest that a congestion-control protocol designed for a simplified model of the network's structure will experience

[3]Results for Flow 2 and Flow 3 (flows crossing only a single link) were nearly identical between the schemes.

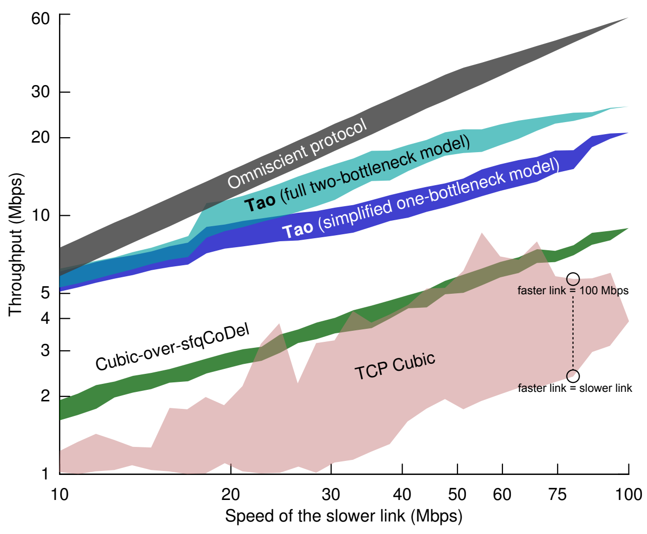


Figure 6: How well do endpoints need to understand the network around them? To assess this, we measure the throughput of four congestion-control schemes across a simulated two-hop network path, as the speed of each link is swept between 10 and 100 Mbps, with 75 ms of delay per hop. A tractable attempt at optimal (Tao) congestion control for a simplified one-bottleneck model of the network performs only a little worse than a protocol designed with knowledge of the network's full two-bottleneck structure.

a small, but quantifiable, penalty to its performance, indicated on the figure as the gap between the two Tao protocols. Results from Cubic and Cubic-over-sfqCoDel are shown for comparison.

These results give some rigorous support to a protocol designer's intuition that understanding the true network in all its complexity may not be crucial.

4.5 Knowledge about incumbent endpoints

We investigated the consequences of designing a congestion-control protocol with the knowledge that some cross-traffic may be the product of pre-existing incumbent protocols.

This question has considerable practical relevance; in practice, the developer of a new network protocol will rarely be able to arrange a "flag day" when all endpoints switch to the new protocol.[4]

On the broad Internet today, cross-traffic will typically be the product of traditional loss-triggered TCP congestion-control protocols, such as NewReno or Cubic. This status quo presents a serious problem for new protocols that seek to perform differently or that try to avoid building up standing queues inside the network.

Some protocols, such as Vegas [6], perform well when contending only against other flows of their own kind, but are "squeezed out" by the more-aggressive cross-traffic produced by traditional TCP. Conventional wisdom is that any "delay-based" protocol will meet a similar fate. This has contributed to a lack of adoption of Vegas and other delay-based protocols.

Ideally, a newly-designed protocol would perform well (e.g. high throughput, low delay) when interacting with other endpoints running the same protocol, **and** would appropriately share a network with incumbent endpoints running traditional TCP. But what are the actual consequences of building this "TCP-awareness" into a protocol?

[4]There has not been a "flag day" on the Internet since the switch to IP in 1983!

Tao	**Link rates**	**RTT**	**Senders**	ON/OFF time
TCP-aware	9–11 Mbps	100 ms	2 Tao 1 Tao, 1 AIMD	5 sec ON/OFF 5 sec ON, 10 ms OFF
TCP-naive	9–11 Mbps	100 ms	2 Tao	5 sec ON/OFF 5 sec ON, 10 ms OFF

(a) Tao protocols with and without TCP awareness, dumbbell network with buffer size 2 BDP

Link rates	**RTT**	**Senders**	ON/OFF time
10 Mbps	100 ms	2 TCP-aware	5 sec ON, 10 ms OFF
10 Mbps	100 ms	2 TCP-naive	5 sec ON, 10 ms OFF
10 Mbps	100 ms	TCP-aware, AIMD	5 sec ON, 10 ms OFF
10 Mbps	100 ms	TCP-naive, AIMD	5 sec ON, 10 ms OFF
10 Mbps	100 ms	2 AIMD	5 sec ON, 10 ms OFF

(b) Testing scenarios for "knowledge about incumbent endpoints" experiment, dumbbell network with buffer size 2 BDP

Table 6: Scenarios for "knowledge about incumbent endpoints" experiment

We studied this by designing two Tao protocols, *TCP-Naive* and *TCP-Aware*, for a simple network—one whose model specified that the cross-traffic would be from the same protocol, and one whose model included a training scenario where the cross-traffic was from traditional TCP half the time (Table 6a). Remy uses an AIMD protocol similar to TCP NewReno to simulate TCP cross-traffic.

The results are shown in Figure 7. In the left panel, protocols compete only with cross-traffic from the same protocol. In this homogeneous setting, adding TCP-awareness to a Tao protocol builds up standing queues, more than doubling the queueing delay without affecting throughput.

But in a mixed setting (Figure 7, right panel) where a Tao competes against TCP NewReno, the TCP-naive Tao is squeezed out and does not get its fair share of the link. In the shaded region showing the results when NewReno competes with the TCP-aware Tao, TCP-awareness allows the Tao to claim its fair share of the link and reduces the queueing delay experienced both by itself and by TCP NewReno.

To further understand the differing performance of TCP-naive and TCP-aware Tao protocols, we inspect their transmissions in the time domain, when contending with a contrived model of TCP cross-traffic (Figure 8) that turns on exactly at time t=5 seconds and turns off exactly at t=10 seconds. The results show that TCP-awareness is a complicated phenomenon, which yields higher delays in isolation but smaller delays when contending with TCP. It is not simply a question of "more aggressive" or "less aggressive" congestion-control.

The results suggest that new delay-minded protocols *can* avoid being squeezed out by traditional loss-triggered TCP, but building in this behavior comes at a cost to performance in the absence of TCP cross-traffic.

4.6 The price of sender diversity

We further generalize the notion of TCP-awareness by asking whether it is possible to design multiple new congestion-control protocols to simultaneously achieve differing objectives when contending on the same bottleneck link.

We consider two extreme cases: a throughput-sensitive sender that weighs throughput ten times over delay ($\delta = 0.1$, *Tpt. Sender*), and a delay-sensitive sender that weighs delay ten times over

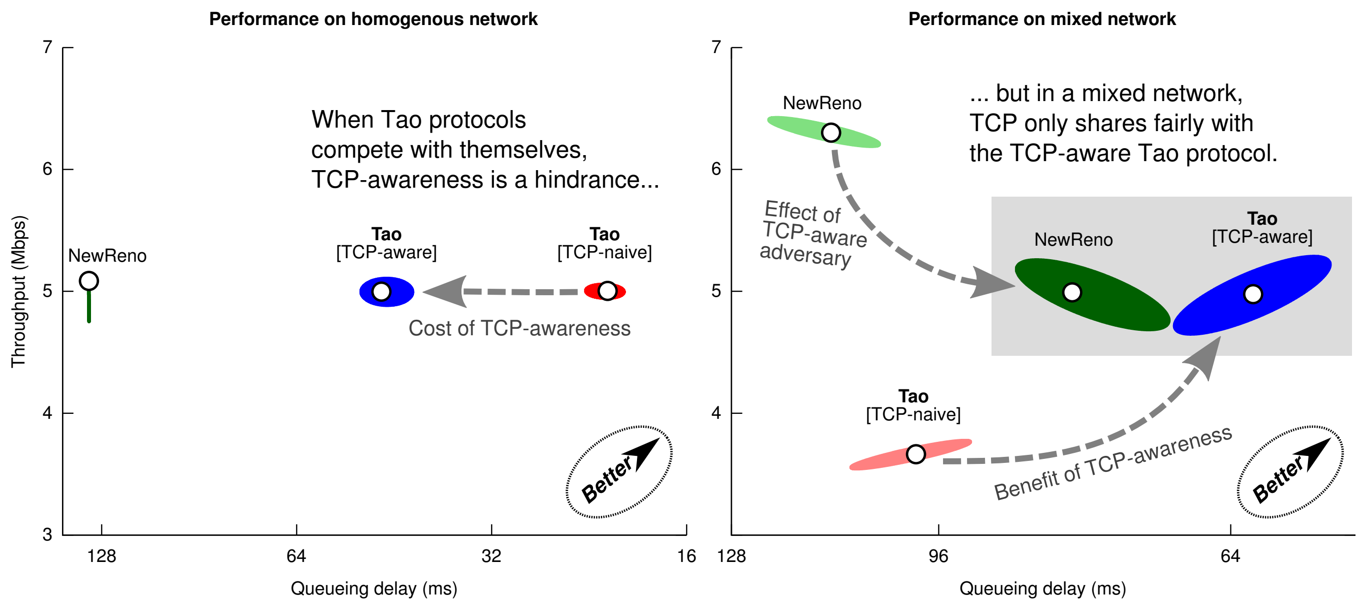


Figure 7: Tao protocols designed with and without TCP-awareness, competing against themselves or against TCP. Shown here, two endpoints contending for a 10 Mbps link with 100 ms RTT, 250 kB of buffer capacity (200 ms maximum queueing delay), and almost-continuous offered load. The fair share of throughput is 5 Mbps per endpoint. Ellipses show 1-σ range of results.

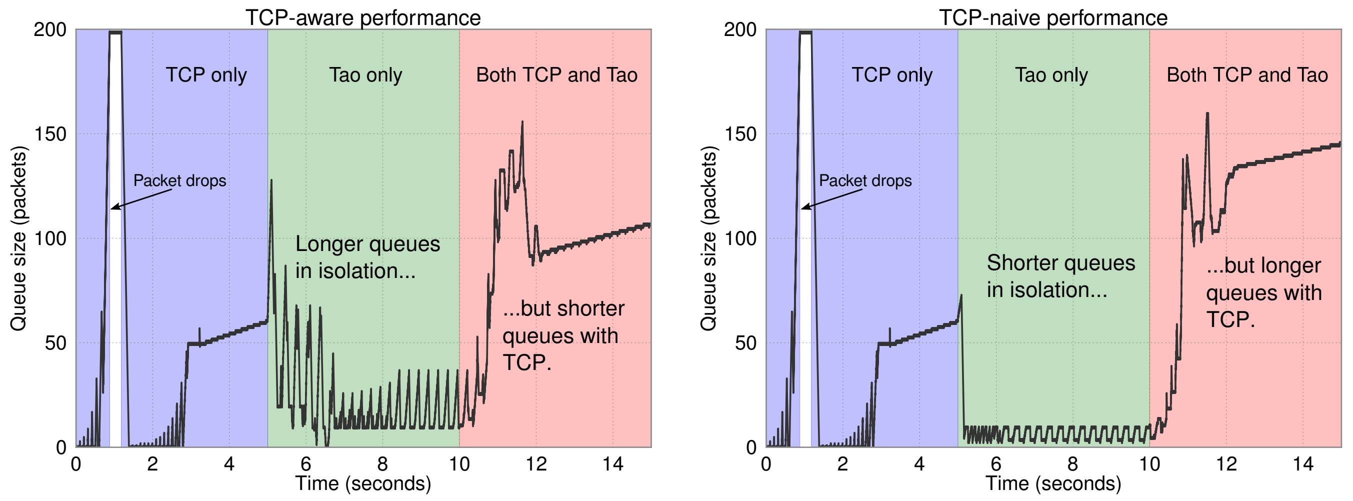


Figure 8: In isolation, a TCP-aware Tao protocol is more aggressive than required, leading to higher delays. But when competing with TCP cross-traffic, the Tao protocol achieves the reverse situation—lower queueing delay, and higher throughput, than the TCP-naive Tao protocol.

throughput ($\delta = 10.0$, *Del. Sender*). Tables 7a and 7b list the training and testing scenarios for this experiment.

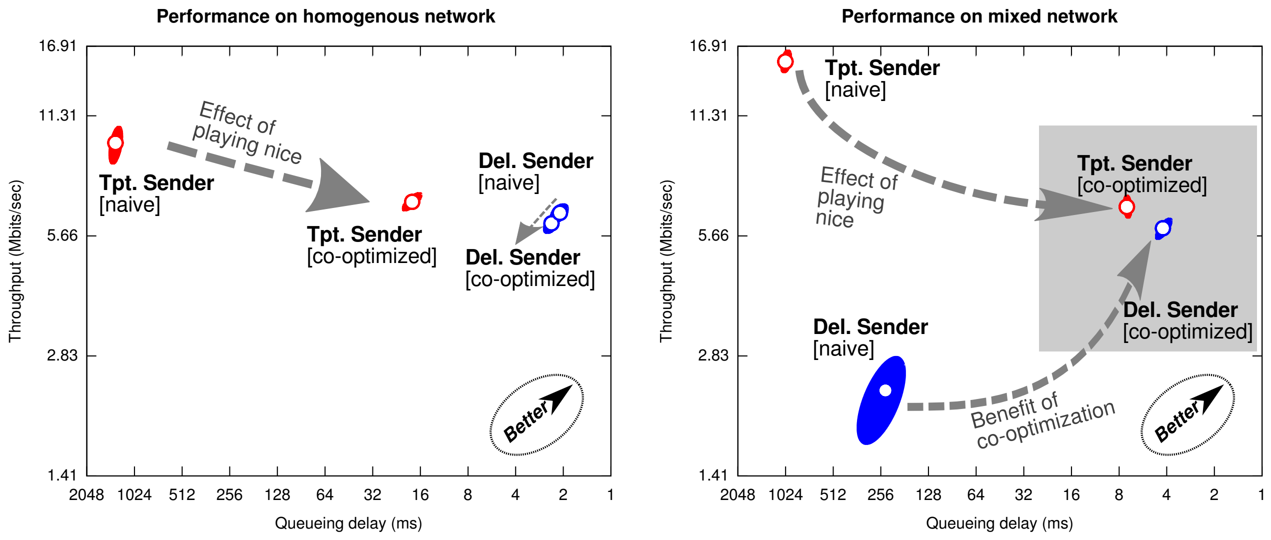
We suspected that achieving diversity in this manner may not be possible, reasoning that endpoints that share the same bottleneck link will necessarily experience the same queueing delay, and therefore endpoints that achieve an optimal throughput-to-delay tradeoff will experience the same overall performance on both throughput and delay.

However, contrary to this hypothesis, our findings are that it is possible to co-optimize congestion-control algorithms such that they each try to obtain a different objective, and yet—because of variable duty cycles—are able to coexist. Even when running together (Figure 9b), the delay-sensitive sender sees lower delay than the throughput-sensitive sender, while the opposite is true with throughput. When congestion-control protocols are co-optimized, but each sender runs homogeneously (Figure 9a), each sender receives higher throughput or lower delay, as the case they may be.

However, coexistence does come at a price to the throughput-sensitive sender, which suffers a loss in throughput by being "nice" to the delay-sensitive sender, both when run with the delay-sensitive sender and when running by itself . By comparison, the performance of the delay-sensitive sender is affected only modestly.

5. Conclusion

This study represents an early step towards a rigorous understanding of the tradeoffs and compromises of network protocol design. We asked: *how easy is it to "learn" a network protocol to achieve desired goals, given a necessarily imperfect model of the networks where it will ultimately be deployed?*



(a) When senders with different preferences run by themselves, training to coexist hurts the throughput of the throughput-sensitive sender. The delay-sensitive sender isn't materially affected.

(b) When senders that are independently optimized for different preferences run on the same network, the delay-sensitive sender experiences much higher delays (compared with the case where it runs by itself). In contrast, co-optimizing the senders hurts the throughput-sensitive sender, but allows the delay-sensitive sender to achieve both lower delay and higher throughput.

Figure 9: The cost and benefits of sender diversity.

Tao	**Link rates**	**RTT**	**Senders**	ON/OFF time	δ	Buffer
Del. Sender	10 Mbps	100 ms	0, 1, or 2 of each type	1 sec ON/OFF	10.0	No drop
Tpt. Sender	10 Mbps	100 ms	0, 1, or 2 of each type	1 sec ON/OFF	0.1	No drop

(a) Tao protocols for diverse senders: one favors throughput, the other delay

Link rates	**RTT**	**Senders**	ON/OFF time	Buffer
10 Mbps	100 ms	1 Del. Sender 1 Tpt.Sender	1 sec ON/OFF	No drop

(b) Testing scenarios for "price of sender diversity" experiment

Table 7: Scenarios for "price of sender diversity" experiment

We investigated several questions under this umbrella by using Remy as a design tool to produce a tractable-attempt-at-optimal (Tao) protocol under various prior assumptions about the network environment. We found only weak evidence of a tradeoff between operating range and performance, even when the operating range covered a thousand-fold range of link speeds. We found that it may be acceptable to simplify some characteristics of the network—such as its topology—when modeling for design purposes. In contrast, features such as the degree of multiplexing and the aggressiveness of contending endpoints were important to capture.

Much remains to be understood about the protocol-design problem before computer-generated protocols will be practically deployable across the broad Internet. For example, can we tractably synthesize a single computer-generated protocol that outperforms human-generated incumbents over a wide range of topologies, link speeds, propagation delays, and degrees of multiplexing simultaneously? Does such a protocol still perform well on networks more complicated than the two-hop parking lot? While our experimental results suggest qualitatively that Remy-generated protocols do not carry a substantial risk of catastrophic congestion collapse, can a protocol optimizer maintain and verify this requirement mechanistically, as part of the design process? Our findings from this study suggest that optimization tools may be able to tackle these kinds of questions in the near future.

Our formalization of the protocol-design process rests on asking the Remy tool to construct a congestion-control scheme, given stated assumptions about the network and an objective function. By contrast, for a human-designed protocol (e.g., existing flavors of TCP), it is usually not possible to describe exactly the implicit underlying assumptions about the network or the intended objective. For that reason, it cannot be guaranteed that our conclusions are truly applicable to the problem of protocol design generally, rather than simply to Remy and similar computer-generated protocol-design methods.

Nonetheless, we do qualitatively observe that the performance of human-designed protocols can depend strongly on network parameters. The single-peaked performance of Cubic-over-sfqCoDel in Figure 4 and the declining performance in Figure 2 suggest that this protocol rests on assumptions about the network parameters—assumptions that may hold more or less well in practice. We therefore believe that our results, and our method generally, can provide useful insights to network protocol designers about what factors are important to model accurately and what factors may be simplified as part of a design process.

In the future, we envision network protocols will be developed mechanistically from first principles: with human designers documenting the objectives and assumptions that a new protocol will have, and automatic processes synthesizing the resulting protocol itself. Such a methodology would allow for a more agile network architecture, because changing requirements or subnetwork behaviors could be accommodated simply by changing the inputs to an automatic process.

While it may have been more straightforward in the past and present for human designers to create protocols directly, based on intuitive design guidelines, the experience of other fields of engineering—mechanical, electrical, civil—suggests to us that protocol design will eventually also adopt a more structured foundation. We will need answers to questions like the ones in this study to make such a transition successful.

6. ACKNOWLEDGMENTS

This work has benefited immensely from discussions with many people, including Scott Shenker, John Wroclawski, Leslie Kaelbling, Chris Amato, Dina Katabi, Frans Kaashoek, Hariharan Rahul, Jonathan Perry, Eugene Wu, and Arun Chaganty. Our shepherd, Aditya Akella, and the anonymous SIGCOMM reviewers gave many helpful comments. We thank Frans Kaashoek, Nickolai Zeldovich, and Li-Shiuan Peh for the use of multicore machines at MIT, and the members of the MIT Center for Wireless Networks and Mobile Computing (Wireless@MIT), including Amazon.com, Cisco, Google, Intel, Mediatek, Microsoft, ST Microelectronics, and Telefonica, for their support. This work was also supported in part by NSF grant CNS-1040072.

REFERENCES

[1] Aspera - High-speed File Transfer Technology - Asperasoft. http://asperasoft.com/technology/.

[2] sfqCoDel. http://www.pollere.net/Txtdocs/sfqcodel.cc.

[3] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data Center TCP (DCTCP). In *SIGCOMM*, 2010.

[4] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4):819–840, Nov. 2002.

[5] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[6] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *SIGCOMM*, 1994.

[7] D.-M. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.

[8] N. Dukkipati and N. McKeown. Why flow-completion time is the right metric for congestion control. *SIGCOMM Comput. Commun. Rev.*, 36(1):59–62, Jan. 2006.

[9] W. Feng, K. Shin, D. Kandlur, and D. Saha. The BLUE Active Queue Management Algorithms. *IEEE/ACM Trans. on Networking*, Aug. 2002.

[10] S. Floyd. TCP and Explicit Congestion Notification. *CCR*, 24(5), Oct. 1994.

[11] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Trans. on Networking*, 1(4), Aug. 1993.

[12] S. Ha, I. Rhee, and L. Xu. CUBIC: A New TCP-Friendly High-Speed TCP Variant. *ACM SIGOPS Operating System Review*, 42(5):64–74, July 2008.

[13] O. Habachi, Y. Hu, M. van der Schaar, Y. Hayel, and F. Wu. Mos-based congestion control for conversational services in wireless environments. *Selected Areas in Communications, IEEE Journal on*, 30(7):1225–1236, August 2012.

[14] J. C. Hoe. Improving the Start-up Behavior of a Congestion Control Scheme for TCP. In *SIGCOMM*, 1996.

[15] V. Jacobson. Congestion Avoidance and Control. In *SIGCOMM*, 1988.

[16] D. Katabi, M. Handley, and C. Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. In *SIGCOMM*, 2002.

[17] F. P. Kelly, A. Maulloo, and D. Tan. Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[18] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. In *SIGCOMM*, 2001.

[19] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In *MobiCom*, 2001.

[20] P. E. McKenney. Stochastic Fairness Queueing. In *INFOCOM*, 1990.

[21] K. Nichols and V. Jacobson. Controlling Queue Delay. *ACM Queue*, 10(5), May 2012.

[22] K. Nichols and V. Jacobson. Controlled Delay Active Queue Management. Technical report, Internet-draft draft-nichols-tsvwg-codel-01, 2013.

[23] R. Pan, B. Prabhakar, and K. Psounis. CHOKe—A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation. In *INFOCOM*, 2000.

[24] K. K. Ramakrishnan and R. Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks. *ACM Trans. on Comp. Sys.*, 8(2):158–181, May 1990.

[25] R. E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[26] R. Srikant. *The Mathematics of Internet Congestion Control*. Birkhauser, 2004.

[27] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A Compound TCP Approach for High-speed and Long Distance Networks. In *INFOCOM*, 2006.

[28] L. G. Valiant. A Theory of the Learnable. *CACM*, 27(11):1134–1142, Nov. 1984.

[29] K. Winstein and H. Balakrishnan. TCP ex Machina: Computer-Generated Congestion Control. In *SIGCOMM*, Hong Kong,, August 2013.

[30] J. Wroclawski. TCP ex Machina. http://www.postel.org/pipermail/end2end-interest/2013-July/008914.html, 2013.

[31] Y. Yi and M. Chiang. Stochastic Network Utility Maximisation. *European Transactions on Telecommunications*, 19(4):421–442, 2008.