Network-Model-Based *vs.* Network-Model-Free Approaches to Internet Congestion Control

Michael Schapira
School of Computer Science and Engineering
Hebrew University of Jerusalem
Israel
schapiram@huji.ac.il

***Abstract*—Congestion control protocol design is an exceptionally complex challenge given the diversity of desiderata and of network environments, as well as the immense breadth of the design space. We will survey below traditional theoretical approaches to this challenge, alongside recently proposed approaches. Our discussion of the limitations and strengths of these approaches will highlight a fundamental distinction: network-model-based *vs.* network-model-free. We will outline research directions that we view as important for building solid foundations for Internet congestion control.**

I. Introduction

Congestion control, the task of modulating the transmission rates of different traffic-sources so as to efficiently utilize network resources, is fundamental to computer networking research and practice. Yet, even after over three decades of research, disagreements linger regarding basic questions such as "*What requirements must congestion control satisfy?*", "*What constitutes a 'good' theoretical model for reasoning about Internet congestion control?*", "*What is the 'right' approach to designing/evaluating congestion control protocols?*".

As applications become more and more demanding (live video, AR/VR, edge computing, IoT, etc.), and the number of network users steeply rises, answering the above questions is ever more important. Indeed, the past few years have witnessed a surge of interest in both industry and academia in revisiting traditional paradigms for congestion control and in exploring new paradigms [2], [5], [8], [9], [26], [27]. The stark conceptual and technical differences between recent proposals for next-generation congestion control evidence the complexity and breadth of this space.

Our aim is to discuss the limitations of traditional theoretical approaches for reasoning about Internet congestion control, highlight crucial *conceptual* differences between recently proposed protocol design frameworks, and identify important directions for further investigation.

A. Why is congestion control so hard?

Consider multiple *connections* (to also referred to as "*flows*" and "*senders*" hereafter) sharing a *single* link, as illustrated in Figure 1. Each sender constantly experiences feedback from its receiver in the form of acknowledgements (ACKs) for received packets and adjusts its transmission rate in response. The manner in which the sending rate is adjusted is determined

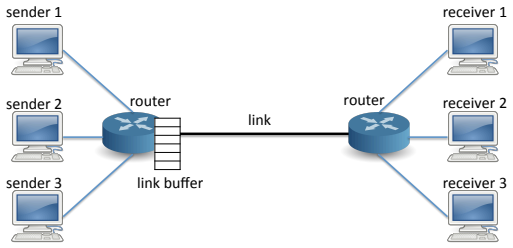

Fig. 1. Connections sharing a single link

by the *congestion control protocol* employed by the two endpoints of the connection according to local observables, such as packet timeouts and round-trip times (RTTs), which, in turn, reflect the network dynamics resulting from the interaction between connections, the link's capacity (bandwidth-delay product), the link buffer's size and packet-queueing policy (e.g., droptail, RED), etc.

Reasoning about the appropriate objectives and conceptual framework for congestion control protocol design is complicated, even in this simple scenario, by the following:

The diversity of desiderata. Different connections might have different *local* performance objectives (reflecting, e.g., small *vs.* large traffic demands, latency- *vs.* bandwidth-sensitive applications). In addition, a protocol designer might wish to satisfy *global* requirements such as optimizing a notion of global performance, quick convergence to a steady state for persistent connections, fairness between competing flows employing the *same* protocol, and friendliness towards competing flows employing *legacy* protocols (e.g., TCP).

The diversity of network environments, which span intra- and inter-datacenter networks, ISP networks, mobile networks, satellite networks, and more. Network environments greatly vary in terms of available bandwidth, loss patterns, latency, level of competition over bottleneck links, etc. In practice, in contrast to the above single link example, the same connection

might be routed through different network segments that exhibit extremely different characteristics.

Uncertainty about the network environment. With the possible exception of very few highly predictable network environments, congestion control decisions are oblivious to, e.g., the number of other connections competing over network resources, at what times these connections enter and leave the network, which congestion control protocols are utilized by other connections, which queueing policies are realized by in-network devices (routers, switches), how underlying routing configurations change over time, etc.

The breadth of the design space. As exemplified by the plethora of (old and new) congestion control schemes, different protocols might rely on different forms of feedback from the traffic receiver and from in-network devices (e.g., packet loss, RTT, packet inter-arrival times, packet marking a la ECN, etc.), might adjust sending rates via different mechanisms (e.g., window-based, pacing), and might operate at different time scales (e.g., per ACK *vs.* per RTT).

B. What is the "right" approach to protocol design?

What can we assume/learn about the network? What can we optimize for? Traditional theoretical frameworks, such as Network Utility Maximization (NUM) [12], [13] and control-theoretic approaches [18], typically rely on explicit or tacit premises about the stability and predictability of the network, e.g., that flows persist "sufficiently long" for rates to converge to a stable global configuration, knowledge of the packet-queueing policies at *all* in-network devices, knowledge of the congestion control protocols employed by *all* connections, etc. We will illustrate this below through NUM.

When designing a congestion control protocol, fully characterizing the network environment in which the protocol is intended to be used is likely infeasible. As will be discussed below, recently proposed frameworks for protocol design differ in their answers to the following fundamental questions: What are reasonable *a priori* assumptions about the network environment? What network parameters can be inferred *online*?

Closely linked to these questions is the question of what global objectives congestion control should *optimize*, as reasoning about global optima often involves nontrivial assumptions about the network, e.g., that network dynamics eventually converge to a *steady state*.

Two design philosophies. Recent approaches to congestion control protocol design can be classified into two categories:

- **Network-model-based** approaches rely on an explicit representation of the network environment, either assumed a priori or inferred online, which supports prospective assessment of how network dynamics evolve and of how different choices of rates will impact performance.
- **Network-model-free** approaches treat the network as a *black box* that translates input rates into output performance rewards, and seek *empirically "good"* mappings between choices of rates and resulting performance.

This distinction resonates the classical division of reinforcement learning (RL) methods into model-based and model-free (see, e.g., [1], [7]).

As will be argued below, **the network-model-based approach and the network-model-free approach to protocol design differ not only in terms of design principles, but also in terms of design goals, and even in what objectives the designer views as achievable**.

We will argue that network-model-based approaches and model-free-approaches have different strengths and limitations: network-model-free approaches are typically better suited for complex network environments and inherently more robust to changes in the network environment; network-based approaches, in contrast, provide better performance when the (crafted a priori and/or inferred) network model is both tractable and accurate. We posit that acquiring such a model for many network environments of interest (e.g., intra-datacenter, "last mile") could be challenging, if not infeasible.

To illustrate these points, we will contrast two recently proposed frameworks for congestion control protocol design that reflect the above two high-level approaches: the *Remy* offline optimization framework [26] and the *Performance-oriented Congestion Control (PCC)* online-learning framework [8], [9].

Last, we will discuss high-level criticisms of network-model-based and network-model-free approaches, and possible responses to these criticisms. We believe that providing a solid theoretical foundations for Internet congestion control involves identifying where each of the approaches is suitable and exploring the shades of gray between purely model-based and purely model-free. We leave the reader with interesting research questions along these lines.

C. Organization

We present the NUM framework in Section II and illustrate, through NUM, the limitations of traditional theoretical approach to congestion control protocol design. The network-model-based Remy framework and the network-model-free PCC online-learning framework are presented in Sections III and IV, respectively. We contrast network-model-based and network-model-free approaches in Section V and discuss interesting directions for future research in Section VI. We briefly discuss in Section VII three examples of non-Remy-generated network-model-based protocols: Sprout [27], BBR [5] and Copa [2]. We conclude in Section VIII.

II. Limitations of Traditional Theoretical Approaches

In landmark publications from the late 1990s, Kelly *et al.* [12], [13] introduced the Network Utility Maximization (NUM) framework for distributed network resource allocation. Since its inception roughly two decades ago, NUM has generated substantial theoretical investigation, leading to the design of new TCP protocols, and shedding new light on existing TCP protocols. NUM inspired recent congestion control schemes such as Remy [26] and Copa [2] (see also NUMfabric [17]).

To simplify exposition, we focus on the following basic, yet expressive, fluid-flow model. n connections, $1, \ldots, n$, send traffic over an undirected *capacitated* network graph $G = (V, E, c)$, where the vertex set V represents routers, the edge set E represents IP-level communication links, and $c : E \rightarrow R$ is a function that assigns a real value to each edge, representing its capacity. Each connection i's traffic traverses a *fixed* network path $P_i \subseteq E$.

Connection i has *utility function* $u_i : [0, M] \rightarrow R$, where M represent a maximum sending rate. The utility function maps each possible sending rate $x_i \in [0, M]$ to a real value. **Importantly, the utility function in NUM is not intended to capture connections' actual local performance objectives. Instead, the utility function (or, more accurately, the combination of all utility functions) is used to specify the *global* objective of the protocol designer, as discussed next.** The optimization objective in NUM is to assign sending rates to connections so as to optimize the sum of derived utility values (referred to as social welfare in economic literature), while not exceeding link capacities. This is formally captured as follows.

$$maximize \ \Sigma_i u_i(x_i)$$

$$subject \ to$$

$$\Sigma_{i|e \in P_i} x_i \leq c(e) \ for \ all \ e \in E$$

$$0 \leq x_i \leq M \ for \ all \ i \in [n]$$

While the above is a *centralized* optimization problem, Kelly *et al.* [13] show that when all utility functions are concave, not only can an optimal solution be efficiently computed, as the above becomes a convex optimization problem, but its solution can be *decentralized* in a natural and elegant way. More specifically, a (Lagrange) duality approach can be leveraged to show that the optimum is reachable by having (1) each edge (link) repeatedly update a local "price" ω_e based on its locally observable load, and (2) each connection i repeatedly update its rate x_i so as to optimize its derived utility under the current link prices, i.e., choose x_i to maximize the expression $u_i(x_i) - x_i \Sigma_{e \in P_i} \omega_e$.

NUM and its extensions have been applied to model a broad variety of network resource allocation problems [28], enabling researchers to reason about decentralized protocols operating at different layers as jointly optimizing a global objective [6].

When approached from a TCP protocol design perspective, the derived link-layer price-adjustment algorithms can be regarded as specifying the queuing policies at routers, with prices representing link loss rates or latencies, and the connection-level algorithms can be viewed as specifying end-to-end congestion control protocols. NUM has proven remarkably useful for both reverse-engineering formal models for preexisting TCP protocols and for designing new, theory-informed TCP protocols.

Reverse engineering TCP protocols. TCP protocols designed in the 1980s and 1990s were based on careful and clever engineering, but lacked solid theoretical foundations. NUM sometimes enables us to reinterpret these protocols as tacitly optimizing a convex optimization problem of the above form. E.g., when $u_i(x_i) = arctan(x_i)$ for each i, and link prices are interpreted as packet loss rates resulting from droptail or RED queueing at routers, the induced dynamic system can be viewed as approximating TCP Reno dynamics. When utility functions are of the form $u_i(x_i) = log(x_i)$, and link prices are interpreted as link latencies under droptail queueing, the resulting system approximates the behavior of TCP Vegas [15].

Designing new TCP protocols. Plugging into NUM choices of utility functions that capture the protocol designer's global optimization objective yields a specification of end-to-end congestion control protocols and of queueing policies at routers that jointly approximate this global optimum [24]. A popular choice of utility function is $u_i(x_i) = \frac{x_i^{1-\alpha}}{1-\alpha}$. The NUM solution for different choices of α, termed the "α-*fair solution*" [16], corresponds to different well-studied notions of fairness for different choices of α: as α goes to 0, the optimum in the limit coincides with total-throughput maximization, as α goes to 1 the optimum coincides with proportional fairness, and as α goes to infinity the optimum coincides with maxmin fairness [16].

What do NUM-generated protocols assume about the network? Are these assumptions reasonable? The basic formulation of NUM, as described above, makes fairly strong assumptions regarding the network: the number of connections is *fixed*, connections have infinite amounts of data to send (so called "elastic traffic"), *all* connections and routers follow the prescribed behavior, and the topology and link capacities of the network are *static*.

While these assumptions render the optimization problem tractable, real-world environments are typically significantly more complex, with connections entering and departing at different times, different connections employing different congestion control protocols, different routers employing different queueing policies, occasional changes to network topology and routes, and so on.

Extensions of the basic NUM framework address this issue through the incorporation of stochastic dynamics into the NUM formulation, e.g., modeling flow arrival and departure as a probabilistic process [28]. While uncertainty about the network might be somewhat alleviated in this manner, skeptics of the NUM approach posit that the sheer complexity of real-world networks transcends the range of network conditions captured by the resulting network models, rendering such models problematic. We will revisit this point in Section V.

We next present two recently proposed approaches to protocol design that contend with uncertainty about network conditions in two fundamentally different manners: Remy [26], which relies on stochastic modeling of the network, and PCC [8], which treats the network as a black box and employs online learning to repeatedly adapt the sending rate in the direction, and to the extent, that yield *empirically* high performance.

III. Remy: Optimizing within a Stochastic Network Model

Similarly to NUM, Remy [26] is an offline optimization framework. Remy takes as input *explicit assumptions* about the network, such as specified ranges of wire speeds, network latency, number of senders on bottleneck links, etc., and also the designer's *global optimization* objective. Remy then generates a *stochastic* model of the network and seeks, within this model, a "good" mapping from observables (e.g., average of ACK inter-arrival times, ratio of current RTT and the minimum observed RTT, etc.) to control actions (such as a multiplier/increment to the congestion window).

Remy-generated protocols are solutions to the following optimization problem:

Input:

1) **A stochastic model of network conditions.** Remy treats the network as drawn from a stochastic generative process. Networks in Remy might be parametrized, e.g., in terms of the number of senders contending over bottleneck links, the bandwidths of bottleneck links, the end-to-end non-queueing latency of network paths, and in-network queue sizes, where each of these parameters is drawn according to an input probability distribution over a given range of possible values. To simplify reasoning about the network, the processes generating the network conditions are assumed in [26] to be *Markovian*, i.e., the network transitions between *states* (each specifying, e.g., the number of connections contending over each bottleneck link, the number of packets in each queue, etc.), and transition to the next network state is dependent only on the current network state. When connections send traffic and when they are silent is also modeled in Remy as a stochastic process.
2) **A global optimization objective.** Similarly to NUM, the input in Remy consists of a global objective that the protocols designer aspires to optimize. Since the network conditions in Remy are modeled as generated by stochastic processes, this global objective is expressed in terms of *expected* values, e.g., the expected throughput (number of bytes sent) and experienced RTT for each connection i, x_i and y_i, respectively. The protocol designer might, for instance, be interested in optimizing a global objective of the form $\Sigma_i log(x_i) - clog(y_i)$ for some constant $c > 0$ (which can be regarded as an adaptation of proportional fairness [13], [16]).
3) **The protocol design space.** A congestion control protocol maps a domain of observables (e.g., exponentially-weighted moving averages of ACK inter-arrival times, the ratio between the most recent RTT and the minimum observed RTT [26]) to a domain of permissible changes to the sending rate (e.g., a multiple to the current congestion window, an increment to the current congestion window). The input in Remy includes a specification of these two domains, i.e., of which observables and of which forms of changes to sending rate are considered.

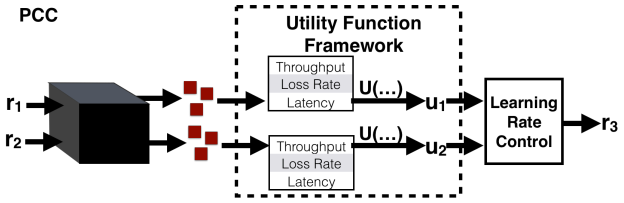

Fig. 2. Illustration of PCC's high-level architecture [9]

Output: A protocol within the design space that, if employed by *all* connections, optimizes the global objective with respect to the stochastic network model.

Remark: Computing the optimal protocol within a stochastic network model of the above described form is formalized in [26] as searching for the best policy with respect to a decentralized partially-observable Markov decision process. However, such optimization is intractable (NEXP-complete [3]), in general. Thus, heuristics for *approximating* the optimum are presented in [26].

What do Remy-generated protocols assume about the network? Similarly to stochastic extensions of NUM [28], the Remy optimization framework reflects two nontrivial assumptions: (1) that network conditions can be faithfully modeled by stochastic processes that are known a priori to the protocol designer, at least to an extent that allows for meaningful optimization, and (2) that the protocol designer can prescribe rate control rules for *all* relevant connections.

IV. PCC: Congestion Control as Online Learning

Recently, congestion control has also been approached from an *online learning* perspective [8], [9] and, more specifically, via the lens of multi-armed-bandit theory (see [4], [11] and references therein). Online learning provides a useful and powerful abstraction for decision making under uncertainty. In the online learning setting, a *decision maker* i repeatedly selects between *strategies* from a fixed *strategy set* S_i. Time is discrete and, at each point in time $t = 1, 2, \ldots, T$, the decision maker selects a strategy $s_i^t \in S_i$. Only *after* selecting strategy s_i^t does the decision maker become aware of the implications selecting that strategy (at that time), in terms of a resulting *utility* value u_i^t. Importantly, the decision maker observes *only* the utility value for the selected strategy and does *not* observe the utility values for other strategies she/he might have selected instead.

Online learning theory contends with questions regarding the achievable *local* guarantees for the decision maker and the achievable *global* guarantees when multiple such decision makers interact. We refer the reader to [4], [11] for relevant background on online learning.

Under the PCC framework [8], [9], Internet congestion control is cast as an online learning task as follows. A sender (the decision maker) repeatedly selects between sending rates

(the strategies) within a certain interval $[0, M]$, where M represents its maximum possible sending rate. Time is divided into consecutive intervals $t = 1, \ldots, T$, termed *Monitors Intervals* (MIs). Each MI is, say, roughly 1-RTT-long, and is devoted to "testing" the implications for performance of sending at a specific rate. After receiving selective ACKs (SACKs) for packets sent at a certain time interval, the sender learns (in *retrospect*) the implications for performance of sending at that rate by translating aggregated statistics (e.g., achieved goodput, packet loss rate, average latency) into a numerical utility value via a *utility function*. **Importantly, unlike the notion of utility function in NUM, the utility function in PCC is intended to reflect the sender's local performance goals.** An online learning algorithm is applied to map the sender's history of sending rates and resulting utility values to its next choice of sending rate. A high-level illustration of the PCC architecture appears in Figure 2.

Designing a PCC protocol entails specifying (1) the utility function, which maps observables from the receiver and the network infrastructure to "utility scores", and (2) the online learning algorithm employed to guide rate selection. We will illustrate below how this is realized in PCC Vivace [9], which borrows ideas from the rich body of literature on online convex optimization [10], [11], [29]. See [9] for more details.

In PCC Vivace, each sender applies a utility function that rewards increase in the connection's throughput and penalizes increase in its loss rate and in its observed "RTT gradient", i.e., the slope of increase/decrease in RTT experienced by the sender within the relevant MI. Vivace's rate control begins with a **slow start** phase in which the sender doubles its sending rate every MI and permanently exits slow start once its empirically-derived utility value decreases for the first time. Vivace then enters the **online learning** phase in which Vivace applies gradient-ascent on the utility function to adapt sending rates.

Vivace's rate-control algorithm is very appealing from an online optimization theory perspective. Specifically, when utility functions are strictly convex, the following two desiderata are fulfilled. (1) Each sender is guaranteed that employing Vivace is (asymptotically) no worse than the optimal *fixed* sending rate in hindsight, a guarantee termed "*no-regret*" in online learning literature [10], [11], [29]. This is a strong guarantee in that it applies even in *adversarial* environments (and so also under *complete* uncertainty about the environment), but it is limited in the sense that it quantifies performance with respect to the actual history of experienced network conditions and not to the conditions that would have resulted from sending at other rates. (2) When multiple senders share the same link, convergence to an equilibrium point is guaranteed and, moreover, for homogenous connections and the appropriate choice of (universal) utility function, the resulting equilibrium is both efficient (well-utilizes the link) and fair [9].

What do PCC-generated protocols assume about the network? PCC treats the network as a black box. Protocols generated within the PCC framework do not postulate *any* causal relation between choices of rates and resulting performance. Unlike NUM and Remy, PCC protocols do not rely on a priori assumptions on the network (e.g., that bottleneck link bandwidths are drawn from a given distribution) and, unlike BBR [5] (see Section VII), PCC protocols do not explicitly infer network parameters (e.g., the bandwidth of the bottleneck link) online.

What do PCC-generated protocols optimize? Unlike NUM and Remy, PCC protocols do not *explicitly* optimize *any* global objective. Instead, each PCC connection optimizes (e.g., in the no-regret sense in PCC Vivace) a *local* optimization objective, as expressed by its utility function. However, even though PCC protocols do not explicitly optimize a global objective, the equilibria resulting from the interaction of multiple PCC connections can sometimes capture desirable global rate-configurations, as in the single bottleneck link setting [8], [9].

Limitations of no-regret. As no-regret relates performance to the best *fixed* strategy in hindsight, the quality of this guarantee in a dynamic environment depends on the speed at which the protocol minimizes "regret" [11]. If, from an arbitrary starting state, regret vanishes to a desired low value within T time units, then the no-regret guarantee applies relative to the best fixed strategy within *every* T units of time. Empirically, PCC Vivace adapts quickly to changes in network conditions [9].

Of course, a guarantee of near-optimality relative to the best *dynamic* strategy would be even better. However, such guarantees often entail assumptions about the environment, e.g., that the network behavior exhibits high regularity. Vivace reflects the design choice of avoiding such assumptions.

V. Contrasting Network-Model-Based and Network-Model-Free Approaches

NUM and Remy are manifestations of the network-model-based approach to congestion control protocol design. Both design frameworks rely on crafting a network model and seeking the optimal protocol with respect to that model *offline*. Other recently proposed network-model-based schemes such as Sprout [27], BBR [5], and Copa [2], to be discussed in Section VII, also incorporate *online* inference of network parameters (e.g., the bandwidth of bottleneck links).

PCC, in contrast, is network-model-free. PCC protocols do not rely on a network model (either assumed a priori or inferred online). Instead, PCC protocols directly evaluate the implications of rate changes for performance and adjust rates in response.

We next present criticisms of each of these two high-level approaches to protocol design. See [21] for a recent debate between proponents of Remy and PCC for more context. We then discuss how these criticisms might be tackled in a principled manner.

A. Criticisms of the two approaches

Criticisms of network-model-reliant approaches: "There are more things in heaven and earth than are dreamt of in your network models!" [21] Skeptics of the network-model-based approach argue that real-world networks are typically

simply too complex for network models to form a solid foundation for decision making. A priori assumptions about the network, as well as attempts to infer network parameters online, are highly prone to be highly inaccurate, leading to misguided protocol designs. In addition, even if a network-model-based protocol performs well in a certain network environment, why would one expect it to continue performing well should the network environment evolve and change? Network-model-reliant approach are thus regarded by critics as both relying on shaky models and as *inherently* non-robust.

Criticisms of network-model-free approaches: "Local optimization = lousy outcomes" [21]. Under PCC, each sender *selfishly* optimizes a locally perceptible utility function that captures its local performance goals. Skeptics of the online learning (and, more, broadly network-model-free) approach to protocol design doubt that any such self-optimizing congestion control scheme is likely to yield desirable global outcomes. A disbeliever in this approach hypothesizes in [21] that not only the global rate configurations reached by PCC-like schemes need not, in general, yield high performance or fairness, but the resulting "rate-configuration equilibria" might vary widely, even for a static network, depending on connections' initial rates or on which connection starts sending first. See articulation of this hypothesis as "Burr's Conjecture" in [21].

B. When is each design approach suitable?

Model-based approaches perform well when provided accurate models. Consider the contrived scenario that a congestion control protocol designer is designing a protocol for the context of a single connection sending alone on a 100Mbs link that never fails. Clearly, the optimal protocol for this setting starts sending at a rate of 100Mbps upon initialization and maintains this rate so long as the connection is active.

Both NUM and Remy, if provided an accurate model of the link and a global objective that rewards increase in throughput and decrease in latency, will generate a protocol that behaves in exactly this manner. In contrast, any network-model-free approach, such as PCC, lacking prior knowledge of the link's bandwidth, is bound to "waste time" on rate exploration prior to converging to the optimal rate.

In general, when relying on an accurate (and tractable) network models, network-model-based approaches are expected to outperform network-model-free approaches in two distinct senses: (1) avoiding the rate-exploration network-model-free schemes apply to associate sending rates with experienced performance, and (2) explicitly optimizing a prescribed global objective when *all* connections are guaranteed to employ the prescribed protocol (as assumed in NUM and Remy).

Not surprisingly, however, optimizing with respect to an inaccurate network model can harm performance [8], [22]. **When can an accurate (at least to the extent needed to provide meaningful performance guarantees) and tractable model of the network be acquired?** While some environments (e.g., arguably, inter-datacenter networks) exhibit high regularity, accurate models for other environments (e.g., "last mile" network segments, intra-datacenter networks) still elude us.

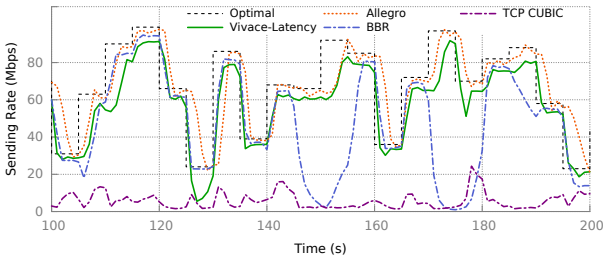

Fig. 3. Protocol performance under varying network conditions [9]

Model-free approaches contend better with uncertainty. Online learning schemes such as PCC Vivace [9] provide a nontrivial guarantee (namely, low regret) for the individual connection even in *adversarial* environments. Thus, network-model-free are inherently better suited for highly unpredictable network environments and also more robust to misguided preconceptions about the network and to unforeseen changes in the network environment [8], [9].

This is illustrated by a recent result from [9], presented in Figure 3, in which the behavior of different protocols is examined on an Emulab [25] link. The RTT, bottleneck bandwidth, and random loss rate on the link all change every 5 seconds, taking values drawn uniformly at random from the ranges 10-100 ms, 10-100 Mbps, and 0-1%, respectively. Observe that PCC Allegro [8] and PCC Vivace [9] closely track the optimal (dashed black) sending rate, whereas TCP Cubic and BBR [5] often send at suboptimal rates. A closer inspection reveals that poor choices of rates are derived from false built-in assumptions about the network; TCP Cubic always interprets packet loss as indicating congestion and lowers the rate drastically in response, whereas BBR, which attempts to estimate the bottleneck of the bandwidth link, exhibits erratic behavior when latency jitters.

As discussed in Section IV, even though network-model-free frameworks such as PCC do not explicitly optimize a global objective, this does not *necessarily* mean that they result in undesirable global behavior. E.g., under the proper choice of utility function, when multiple PCC connections compete over a single link, convergence to a fair and efficient bandwidth allocation is both provably and empirically guaranteed [8], [9].

Conclusions. The question of whether to apply a network-model-based approach or a network-model-free approach in a specific context boils down to the following two questions: (1) **to what extent need network-model-based protocol designers be able characterize the network conditions to attain high performance?** and (2) **under what conditions do network-model-free approaches yield desirable global outcomes?** Naturally, the answers to these questions vary across network domains. We discuss these, and additional research challenges, next.

VI. Directions for Further Investigation

We next outline interesting directions for future research. We believe that pursuing these directions is important for basing congestion control designs on sound theoretical foundations.

A. In what contexts are network models useful?

Validating the usefulness of network-model-based design frameworks in a specific context requires addressing the following questions: (1) **is an *ideal* network model sufficiently expressive to guide rate-control decisions in this context?** (2) assuming that the answer to the previous question is positive, **how close can the protocol designer come to generating this model, and what are the implications of misrepresenting or misestimating different aspects of the model and the network parameters?**, and (3) **is computing the optimal (or approximately optimal) protocol with respect to the network model feasible?**

One approach for tackling the first of these questions is applying *explainable* and *interpretable* [20] machine learning techniques to empirically-derived network traces in search of tractable models with high predictive power, and then studying the implications of optimizing rate selection with respect to these.

An important first step towards answering the second question is taken in [22], which employs the Remy framework to reason about protocol design given *imperfect* or *inaccurate* prior knowledge about a baseline network model. The findings suggest that while even gross misestimations of some network parameters (e.g., range of link speeds) have low impact on protocol performance, other parameters (e.g., the number of connections on bottleneck links) are important to capture. Providing *theoretical* support for these claims is an important research agenda.

Regarding the third question, as discussed in [26], from a theoretical standpoint, efficiently generating the optimal Remy-generated protocol is, in general, intractable (NEXP-complete). What about generating protocols that provide good approximations to the optimum?

B. Under what conditions do network-model-free approaches yield desirable global outcomes?

Theoretical analyses of online-learning congestion control (PCC [8], [9]) focused on a simple, single-link scenario. Not surprisingly, as discussed in [21], a protocol that exhibits good behavior on a single link might behave erratically on a larger network. This motivates the following theoretical questions: (1) **under what conditions is a connection employing online-learning guaranteed to have low regret?** (2) **under what conditions is convergence to a steady state guaranteed and, more specifically, when is the resulting steady state independent of the initial sending rates?** (3) **under what conditions is convergence to efficient and fair allocations of network resources guaranteed?**

Tackling the second and third questions from a theoretical perspective involves formulating non-cooperative games modeling the interaction of multiple connections in the considered network environment. Indeed, the second question can be formulated as a question regarding the convergence of no-regret game dynamics to equilibria, whereas the third question can be formulated as quantifying the *price of anarchy* [14], [19] in the relevant class of non-cooperative games.

C. Improving upon regret-minimization?

PCC (more specifically, PCC Vivace [9]) leverages insights and machinery from the theory on regret minimization in the multi-armed bandit setting to guide rate control. Can other ideas and techniques from the rich literature on the broader domain of reinforcement learning [23] provide better performance?

One approach for tackling this question is investigating the performance benefits resulting from the application of deep reinforcement learning schemes (which constitute the state of the art in many contexts) to rate selection.

D. Hybrid approaches

Our discussion thus far focused on purely network-model-based approaches and purely network-model-free approaches. However, one can envision ways in which these two approaches can be merged in an effort to attain the best of both worlds (e.g.,to reduce rate exploration while still being robust to network changes).

One such approach might be to initialize the parameters of protocols designed within a network-model-free framework to reflect the designer's (potentially inaccurate) preconceptions about the network environment. This could potentially result in reduced rate-exploration when the designer's mental picture of the network is fairly accurate.

Another approach is to only model some aspects of the network while treating other aspects as a black box. Consider, e.g., the scenario that the protocol is intended to be *ubiquitously* deployed within a single organizational network that exhibits highly unpredictable behavior (e.g., within a datacenter). The protocol designer might derive performance benefits from relying on the assumption that all connections execute the protocol, but might not wish to assume/infer anything about the competition over bottleneck links.

VII. More Network-Model-Based Approaches: Sprout, BBR, and Copa

We described above two general network-model-reliant frameworks for protocol design: the NUM framework (in Section II) and the Remy framework (in Section III). Below, we briefly discuss three specific network-model-based congestion control protocols that do not fall within these frameworks.

Sprout [27]. Sprout uses probabilistic *online* inference for forecasting, based on packet arrival times at the receiver, the bandwidth of the bottleneck link in cellular wireless networks. Sprout uses these short-term forecasts to select rates such that sent traffic is predicted to reach the receiver shortly after

transmission with high probability, while bounding the risk that queueing-induced delays exceed a certain threshold.

BBR [5]. BBR (Bottleneck Bandwidth and RTT) models the network pipe as a single link, representing the bottleneck link on the route. A sender utilizing BBR repeatedly probes the bandwidth and RTT, and paces the rate so as to track the link's bandwidth.

Copa [2]. Copa optimizes rate selection under a *Markovian* packet-arrival model in which packet arrivals at the connection's bottleneck link are modeled as a Poisson distribution. In addition, a Copa-sender attempts to detect in real time when it is in competition with purely-loss-based congestion-controlled connections (such as TCP Cubic).

As exemplified by NUM, Remy, Sprout, BBR, and Copa, network-model-based approaches to congestion control protocol design come in many flavors. Different approaches might differ in which aspects of the network model are assumed a priori and which are inferred online, and in whether the network model is deterministic or probabilistic.

VIII. Conclusion

Congestion control has been fundamental to networking research for decades, and is only expected to grow in importance as the already fierce competition over scarce network resources becomes even fiercer given the overwhelming growth in traffic volumes and applications' ever growing performance demands. We discussed different frameworks for congestion control protocol design and highlighted the conceptual differences between them. We believe that further deliberation of the challenges facing congestion control, approaches to protocol design, and open questions outlined above, could prove conducive to the quest for better Internet congestion control.

Acknowledgment

I thank Brighten Godfrey and Keith Winstein for many insightful discussions that profoundly impacted my views on congestion control. I also wish to thank the past, present, and future members of the PCC team at Hebrew U and UIUC. I am grateful to Costin Raiciu and the HSPR 2018 co-chairs, Giuseppe Bianchi and Theo Benson, for inviting me to present this paper at the conference. I thank Google, Huawei, and the ISF, for ongoing support of the PCC project.

References

[1] Abbeel, P., Quigley, M., and Ng, A. Y. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning* (2006), ICML '06, pp. 1–8.

[2] Arun, V., and Balakrishnan, H. Copa: Practical delay-based congestion control for the internet. In *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018* (2018), pp. 329–342.

[3] Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research 27*, 4 (2002), 819–840.

[4] Bubeck, S., and Cesa-Bianchi, N. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning 5*, 1 (2012), 1–122.

[5] Cardwell, N., Cheng, Y., Gunn, C., Yeganeh, S., and Jacobson, V. BBR: Congestion-based congestion control. *Queue 14*, 5 (2016), 50.

[6] Chiang, M., Low, S. H., Calderbank, A. R., and Doyle, J. C. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE 95*, 1 (2007), 255–312.

[7] Deisenroth, M. P., Neumann, G., and Peters, J. A survey on policy search for robotics. *Found. Trends Robot 2*, 1–2 (August 2013), 1–142.

[8] Dong, M., Li, Q., Zarchy, D., Godfrey, P. B., and Schapira, M. PCC: Re-architecting Congestion Control for Consistent High Performance. *Proc. of NSDI* (March 2015).

[9] Dong, M., Meng, T., Zarchy, D., Arslan, E., Gilad, Y., Godfrey, B., and Schapira, M. PCC vivace: Online-learning congestion control. In *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018* (2018), pp. 343–356.

[10] Even-Dar, M. E., Mansour, Y., and Nadav, U. On the convergence of regret minimization dynamics in concave games. *Proc. of ACM symposium on Theory of computing* (2009).

[11] Hazan, E. Introduction to online convex optimization. http://ocobook.cs.princeton.edu/OCObook.pdf.

[12] Kelly, F. Charging and rate control for elastic traffic. *European Transactions on Telecommunications* (1997).

[13] Kelly, F., Maulloo, A., and Tan, D. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society* (1998).

[14] Koutsoupias, E., and Papadimitriou, C. Worst-case equilibria. In *Proceedings of the 16th Annual Conference on Theoretical Aspects of Computer Science* (Berlin, Heidelberg, 1999), STACS'99, Springer-Verlag, pp. 404–413.

[15] Low, S. H., Peterson, L. L., and Wang, L. Understanding tcp vegas: A duality model. *J. ACM 49*, 2 (March 2002), 207–235.

[16] Mo, J., and Walrand, J. Fair end-to-end window-based congestion control. *IEEE/ACM Trans. Netw. 8*, 5 (October 2000), 556–567.

[17] Nagaraj, K., Bharadia, D., Mao, H., Chinchali, S., Alizadeh, M., and Katti, S. Numfabric: Fast and flexible bandwidth allocation in datacenters. In *Proceedings of the 2016 ACM SIGCOMM Conference* (New York, NY, USA, 2016), SIGCOMM '16, ACM, pp. 188–201.

[18] Paganini, F., Doyle, J., and Low, S. H. *A Control Theoretical Look at Internet Congestion Control*. Springer Berlin Heidelberg, 2003, pp. 17–31.

[19] Roughgarden, T., and Tardos, . *Introduction to the Inefficiency of Equilibria*. Cambridge University Press, 2007, p. 443460.

[20] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall Press, Upper Saddle River, NJ, USA, 2009.

[21] Schapira, M., and Winstein, K. Congestion-control throwdown. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks* (New York, NY, USA, 2017), HotNets-XVI, ACM, pp. 122–128.

[22] Sivaraman, A., Winstein, K., Thaker, P., and Balakrishnan, H. An experimental study of the learnability of congestion control. *Proc. of ACM SIGCOMM* (August 2014).

[23] Sutton, R. S., and Barto, A. G. *Introduction to Reinforcement Learning*, 1st ed. MIT Press, Cambridge, MA, USA, 1998.

[24] Wei, D., Jin, C., Low, S., and Hegde, S. FAST TCP: motivation, architecture, algorithms, performance. *IEEE/ACM Transactions on Networking* (2006).

[25] White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, G., Newbold, M., Hibler, M., Barb, C., and Joglekar, A. An integrated experimental environment for distributed systems and networks. *Proc. of OSDI* (December 2002).

[26] Winstein, K., and Balakrishnan, H. TCP ex Machina: computer-generated congestion control. *Proc. of ACM SIGCOMM* (August 2013).

[27] Winstein, K., Sivaraman, A., and Balakrishnan, H. Stochastic forecasts achieve high throughput and low delay over cellular networks. *Proc. of NSDI* (March 2013).

[28] Yi, Y., and Chiang, M. Stochastic network utility maximisation - a tribute to kelly's paper published in this journal a decade ago. *European Transactions on Telecommunications 19*, 4 (2008), 421–442.

[29] Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *ICML* (2003), AAAI Press, pp. 928–936.