Unleashing the Potential of Data-Driven Networking

Junchen Jiang[†], Vyas Sekar[†], Ion Stoica[⋆+◦], Hui Zhang[†+]

[†]CMU, [⋆]UC Berkeley, [+]Conviva, [◦]Databricks

Abstract. The last few years have witnessed the coming of age of data-driven paradigm in various aspects of computing (partly) empowered by advances in distributed system research (cloud computing, MapReduce, etc). In this paper, we observe that the benefits can flow the opposite direction: the design and management of networked systems can be improved by data-driven paradigm. To this end, we present DDN, a new design framework for network protocols based on data-driven paradigm. We argue that DDN has the potential to significantly achieve better performance through harnessing more data than one single flow. Furthermore, we systematize existing instantiations of DDN by creating a unified framework for DDN, and use the framework to shed light on the common challenges and reusable design principles. We believe that by systematizing this paradigm as a broader community, we can unleash the unharnessed potential of DDN.

1 Introduction

Ask not what networking can do for data;
Ask what data can do for networking!

Most networked applications require some form of control and adaptation logic to respond to changing network conditions. Traditionally, such control and adaptation strategies in application and transport protocols have come to rely on a *per-session* [1] *view* and using *manually designed* strategies. For instance, TCP congestion control relies on many human-picked constants (e.g, init_cwnd, AIMD parameters) and is driven by feedback (e.g., acks, ECN) observed within the current flow. In some sense, this defacto design template has been passed down across generations of designers as a form of conventional wisdom drawn from lessons (e.g., end-to-end principles [50], congestion collapse [28]) from the early days of the Internet.

While this conventional wisdom has indeed served us well for decades, today we face a confluence of use case "pulls" that increasingly make such approaches ineffective:

- First, we see *growing expectations* of user-perceived Quality of Experience (QoE) [31]. Thus, conventional approaches that worked well for a "best effort" mentality are no longer good enough!

[1] A session could be an application session (e.g., video session, web session), or a transport session (e.g., TCP session).

– Second, we observe a *growing decision space* of potential control decisions to optimize application quality. Consequently, trial-and-error strategies driven by single-session feedback are fundamentally inefficient and slow in exploring the decision space and reacting to changes. For instance, it takes a video player several chunks (roughly 10s of seconds) to converge to an optimal combination of CDN and bitrate [7].
– Third, we see an *increasing heterogeneity* in operating conditions, each requiring different control logic and parameters. For instance, TCP parameters such as initial congestion window and AIMD parameters could be tweaked to work better in different operating conditions [57, 20].

Tackling these challenges requires us to radically rethink the design of the control plane of network protocols. In this context, we observe an opportunity for a new paradigm, which we refer to as *Data-Driven Networking or DDN*, inspired in part by the successes in other aspects of computing driven by the ability to collect and extract insights from large corpus of data [16, 25]. DDN transforms the design of the "control plane" of network protocols by the availability of massive data from millions of endpoints and devices. This is orthogonal to recent efforts to optimize the "data planes", such as ICN [29]. More specifically, a DDN-based control loop is driven by real-time *multi-session* (not single-session) view of *in-situ quality* [23] measurement (not active measurements or indirect metrics), and *automatically* tuned actuation algorithms based on data-driven insights with little to no manual tuning. For instance, to apply the concept of DDN to optimize quality for video streaming, one can think of a controller that takes as input the quality measurements of millions concurrent and history video clients, and use such data to automatically learn the best configuration such as server/CDN/cache, bitrate, and network path for each client.

In parallel to the use-case pulls described above, we also observe that DDN fortuitously has key technology "pushes" that are suitably aligned. Specifically, many application providers today have widely deployed client-side instrumentation to collect real-time performance data [23], and "big data" processing capabilities are finally a reality [6].

We are already seeing point evidence of this paradigm shift starting to reach the shores of many networked applications. This includes work in optimizing Internet video streaming (e.g., [32, 24]), Internet telephony (e.g., [30]), cloud service availability (e.g., [39]), data center scheduling (e.g., [56]), congestion control (e.g., [51, 18, 57]), and network management (e.g., [15]), as well as industrial initiatives (e.g., [2, 8]).

To fully unleash the potential of DDN, however, we identify key challenges spanning a wide spectrum: (1) algorithms and machine learning (e.g., tackling high dimensionality, data sparsity, and exploration-exploitation tradeoffs), (2) system design (e.g., balancing scalability, responsiveness and staleness), and (3) broader architectural questions (e.g., stability and interactions with other possibly competing control loops). Unfortunately, the aforementioned early DDN efforts suffer fundamental shortcomings that ultimately undermines the full potential of this paradigm:

– First, each of these efforts offers point solutions tackling a small subset of DDN challenges (tied to specific workload assumptions) and as such does not focus on

generalizable algorithmic or system-design insights for future DDN-based deployments. For instance, can these primitives (e.g., quality-determining features learned by CFA [32] + split control plane from C3 [24] + exploration-exploration strategy from VIA [30]) be composed and if so how? Are these missing any hidden aspects; e.g., how to ensure control stability or handle flash crowds?

– Second, they present techniques under application-specific assumptions and it is unclear if, why, and by how much they could benefit other applications. For instance, CFA assumes that video quality-determining features are persistent, while VIA assumes that VoIP quality distribution between the same source and destination ASes depends only on relay servers, and it is not obvious if these are necessary or sufficient, or how much the benefit would be if we relax these assumptions.

Rather than develop yet-another point solution to a narrow application, this paper takes a *first-principles* approach to help democratize the potential benefits of the DDN paradigm to a broader spectrum of use cases. To that end, this paper makes two contributions:

1. Formalizing DDN (§2). We define key features that distance DDN from alternatives, provide illustrative examples of when DDN benefits, and generalize the design space as well as the use cases of DDN.

2. Common building blocks (§3). We sketch a common platform for socializing the benefits of DDN by developing key building blocks. We articulate four high-level technical challenges, which are still open: how to extrapolate potential benefits of DDN, infer optimal decisions with sparse data, implement real-time feedback loops at scale, and stabilize DDN control behaviors. We also see an opportunity of adapting abstractions and techniques from machine learning and control theory, and we offer promising prescriptive roadmap to address the challenges.

2 Formalizing DDN

In this section, we give a conceptual overview of DDN, contrasting it to traditional approaches. We present early evidence from prior work on the case for DDN. We end this section by identifying key factors that can impact the potential benefits of DDN.

2.1 What is DDN?

DDN is a new paradigm for designing the control plane of network protocols. It consists of two components: (a) the *client-side instrumentation* integrates with the client-side application[2] to measure client-perceived quality and apply decisions made by DDN; and (b) the *DDN controller* runs two loosely coupled steps:

1. Aggregate quality measurement from client-side instrumentation into a *multi-session view* – a representation for summarizing the performance of similar sessions.
2. Make control decisions based on the multi-session views, and send them to client-side instrumentation.

To understand the contrast between DDN and traditional control planes, it is useful to revisit the two logical steps in the workflow of *any* control plane (Figure 1): *sensing*,

[2] We use "client" to denote where a session is actually run.

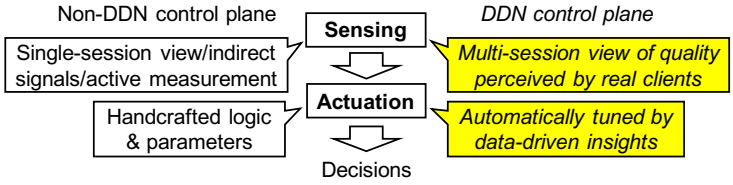

Fig. 1: ***DDN control plane is fundamentally different to non-DDN ones on both sensing and actuation.***

which gives the feedback data to control plane, and *actuation*, which turns the feedback data to control decisions. DDN radically departs from non-DDN designs on both fronts with three definitive features. (Table 1 provides some examples to contrast between DDN and non-DDN designs.)

- **Multi-session view:** Sensing of DDN is based on multiple sessions, rather than single session. By extending the spatial scope of sensing to many sessions, DDN can predict the quality of a decision even before a session actually uses it, as long as the decision has been used by some similar sessions.
- **In-situ quality:** In DDN, what to be "sensed" is exactly what to be optimized, i.e., quality perceived by all historical and ongoing sessions; not indirect signals on quality (e.g., acks [28] or bandwidth [37]), or active probes from a handful of vantage points (e.g., iPlane [42]). While in-situ quality data may compromise on the fidelity of individual measurement, they are far more efficient than alternatives in obtaining a panoramic and representative view of client-perceived quality from growingly diverse platforms [23]. Relying solely on in-situ quality data also serves pragmatic purposes as many application providers today already have a vested interest in measuring user-perceived quality for various reasons [1, 40, 36].
- **Automatically tuned control logic:** To take full advantage of the enriched sensing data, actuation algorithms of DDN should be dynamically tuned by data-driven insights with little to no manual configuration. Unlike today's protocols where hand-picked constants are used as key parameters (e.g., init_cwnd and initial video bitrate), DDN picks parameters [57] and control logic [54] based on quality feedback which indicates what suit the current operating condition the best. Meanwhile, the DDN control logic also needs to handle the *downside* of having more data (e.g., lack of fidelity in client-side measurement, and whether the data source is trustworthy) by harnessing the "unreasonable effectiveness of data" [25].

2.2 Early promise of DDN

Several early applications of DDN from prior work have shown tremendous promise of this new paradigm. They showcase how DDN can be adapted to various use cases to exploit their potential benefits (depicted in Figure 2).

CDN/bitrate selection for video: The first example shows how a global view of video quality can optimize CDN and bitrate selection for individual video sessions. Video players today have the flexibility of streaming content from one of multiple CDNs

	Sensing		Actuation
	Multi-session	*In-situ quality*	*Auto-tuned*
DDN	✓	✓	✓
TCP AIMD [28]	×	×	×
PCC [18]	×	✓	✓
OSPF, BwE [37]	✓	×	×
iPlane [42]	✓	×	NA
RemyCC [57]	×	×	✓

Table 1: ***Difference of DDN to non-DDN strategies.***

and bitrates. However, with only information on a single session, the current protocols always start with a default CDN and fixed (and conservative) bitrate, and gradually converge to a better bitrate and CDN by local trial-and-error strategies. Given both performance of CDNs and client-side bandwidth have a substantial spatial diversity and temporal variability [40], there is a remarkable room for improvement by dynamically mapping a session to the optimal CDN and bitrate with no trial-and-errors. Prior work [24, 32, 54] exploits this opportunity by mapping a video session to the CDN and bitrate that has the best quality on similar sessions (e.g., those in the same AS and watching the same video content); and it can reduce the session duration spent on rebuffering by 50% without lowering bitrates.

Relay selection for Internet telephony: The second example shows how VoIP quality can be improved by a controller that selects relay servers judiciously. VoIP applications (e.g., Hangout and Skype) use relay servers for NAT traversal, where the selection of relay servers has traditionally been agnostic to real-time network conditions. But recent work has shown a substantial room for improvement on call quality by selecting optimal relay servers for each call [26]. For instance, recent work [30] shows that one can select near-optimal relay servers for individual Skype calls by identifying which relay has the best quality for similar calls (e.g., those between the same source and destination ASes on the same date); compared to non-relayed paths, this can alleviate 42% of Skype calls whose quality is impacted by high packet loss rate ($> 1.2\%$).

Online service cluster selection: The last example shows how the quality of online services (e.g., search engines) can be improved by a centralized control platform, which selects optimal proxies by consolidating quality data of multiple applications and profiles of the infrastructure. Recent work [39] takes the stance of a company who has the visibility and controllability over multiple applications as well as key infrastructure building blocks. By measuring end-to-end quality from clients and dynamically modeling the workload of network paths and servers, it can select proxies that reduce mean latency by 60% and carry $2\times$ more traffic, compared with a baseline that finds proxies by Anycast.

2.3 When is DDN beneficial?

A natural question that these case studies raise is the following: When is DDN beneficial? That is, what properties of an application allow it to benefit significantly from

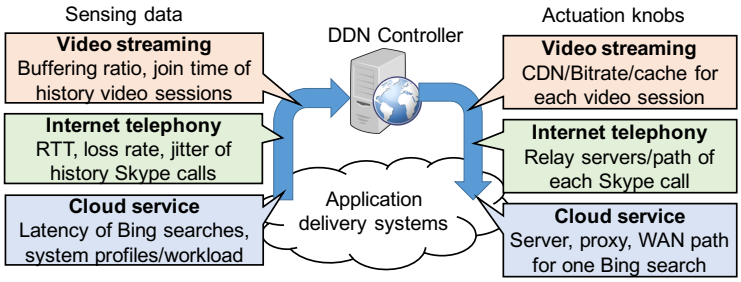

Fig. 2: ***Examples of DDN from prior work.***

DDN. Drawing from this prior work, we highlight key features that impact the potential reward of using DDN.

- **Heterogeneity of decision space:** DDN has great room for improvement, when (1) the quality of each decision has substantial spatial diversity or temporal variability (e.g., same Skype relay cluster has significantly different performance depending on where the caller and callee are and when the call happens [30]); (2) the decision space is fine-grained (e.g., server selection has more room for improvement on video quality than CDN selection [24]); and (3) each decision needs to be probed separately (e.g., each server is reached by different WAN path [39]).
- **Usefulness of feedback data:** How useful the feedback data is in driving DDN control logic depends on whether there are enough sessions for DDN to reliably estimate the quality of candidate decisions.This happens when we have enough data to gain statistical significance and/or measurement noise is low. For instance, network latency naturally is more stable than packet loss rate, so less sessions are needed to find the server with low latency than needed to find the server with low loss rate.
- **Tail vs. median performance:** One of the key takeaways from prior work is that DDN has greater benefits on the tail than on the median, possibly because most applications have been heavily optimized for the "common" clients. For instance, the median buffering ratio (fraction of time spent on buffering per session) of most video providers is zero, while at least 10% sessions suffer from high ($> 1\%$) buffering ratio. It is worth noting that the sessions on the tail are not widely scattered; rather they often represent some groups of users (e.g., from the same small ASes [32]), which makes tail improvement more relevant and critical.

Example of new use case: Now, we apply these features to a new use case where DDN is expected to have a great advantage. Crowdsourced live streaming [9, 5] faces not only the challenges of traditional live streaming, but low delay for interactivity and the need to multicast from any user at any time. It makes a perfect case for DDN, because it has (1) a heterogeneous decision space: performance of different caches and codec servers varies significant across viewers and sources and over time [14]; and (2) useful feedback data: a streaming event typically has over hundreds of viewers, which can be used to explore various decisions.

Challenges	Opportunities
What-if analysis	Doubly robust estimator [19]
High dimensionality vs. data sparsity	Contextual multi-armed bandit [41]
Scale, responsiveness vs. data freshness	Regressograms and localized control logic [49]
Control stability	Benevolent feedback delay [12]

Table 2: ***Summary of key challenges and opportunities***

3 Key challenges and opportunities

In this section, we articulate four high-level challenges that are key to unleashing the full potential of DDN. For a provider, the fundamental questions are: (1) how much will I actually benefit from DDN; (2) are there existing algorithms; (3) how to build the system at scale; and (4) how will it interact with other parts of the Internet. For each challenge, we identify *opportunities* to address them using techniques from machine learning and control theory (Table 2). However, we also observe that it is not a straightforward application of existing techniques from these communities and it requires care to adapt them to handle network- and domain-specific issues (e.g., load effects) as well as to exploit domain-specific opportunities to enable simple-yet-effective designs.

In the interest of clarity, we focus on a specific DDN design point (as in [32, 24, 30]): single-provider, logically centralized, with control logic updated in real time. However, these challenges and opportunities manifest themselves in other forms of DDN as well (§4.1).

3.1 "What-if" analysis

Problem statement: The first question for any application provider (e.g., Netflix) is that before implementing DDN, "can I quantify how much DDN would actually improve my application's quality (e.g., video quality)?"

Limitations of strawmen: Simulation is known to be unrealistic for evaluating wide-area network performance [22]. Small-scale emulation (e.g., Emulab [3]) is more realistic, but lacks the scale needed to reveal the real benefits of DDN. Finally, A/B tests may be used but require that each DDN strategy be evaluated on substantial amount of real clients, which application providers are often reluctant to do [36].

Now, the fact that application providers today have already collected massive amount of quality measurements opens an opportunity for data-driven evaluation, also known as *off-policy evaluation* [44]: given the quality of many sessions whose decisions were driven by some control logic, can we evaluate the quality of a different control logic? This is, however, not straightforward and simple solutions may have either *bias* (due to hidden factors) or high *variance* (when data is too sparse) or both. Suppose the dataset was collected by assigning wireless users to Akamai and cable users to Limelight, any method will falsely claim Akamai has worse quality than Limelight, if the last-mile connection is hidden.

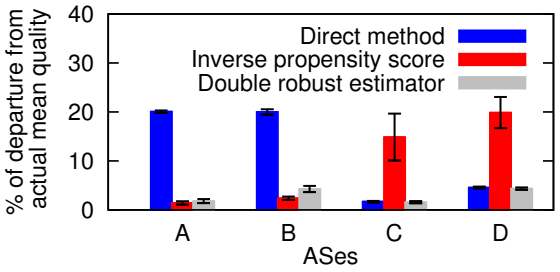

Fig. 3: ***Strawman data-driven evaluation methods have significant evaluation error. Direct method is biased by hidden features (AS A and B), and IPS method has high variance with sparse data (AS C and D). In contrast, the DR estimator achieves the best of two methods.***

These are not merely theoretical concerns; Figure 3 shows how these manifest in real-world datasets for some seemingly natural solutions. We use a dataset of video quality in four ASes. For each AS, we randomly split it into two subsets, and show the difference between the actual quality of one subset and the extrapolation made by two standard off-policy evaluation methods based on the other subset. On one hand, the direct method (e.g., [35]) has low variance, but is vulnerable to bias of hidden features. The inverse propensity score (IPS) (e.g., used in [32, 38]), on the other hand, focuses on the sessions for which DDN makes the same decision as in the dataset, so we know their actual quality, but IPS has high variance when such overlapping is rare. As shown in Figure 3, we see that both direct method and IPS suffer from high evaluation error, due to high bias or variance. Furthermore, it is not easy to check when and where these will fail (e.g., existence of hidden features cannot be easily verified).

Opportunity: Doubly robust estimator. A promising alternative is the recently proposed *doubly robust* (DR) estimator [19, 13], which combines the best of the direct and IPS methods described above. It is unbiased and has low variance, when *either* we know all confounding factors (i.e., suitable for direct method) *or* DDN makes overlapping decisions with the dataset on many sessions (i.e., suitable for IPS). More importantly, we do not need to explicitly know when and which assumption holds, and thus it naturally achieves the best of two methods (e.g., as shown in Figure 3).

While the DR estimator is a promising starting point for reliable data-driven evaluation, there are network-specific factors it does not consider. For instance, the DR estimator will not identify quality degradation due to server overload, if such overload never happens in the dataset. This can be handled by simple techniques (e.g., a load-dependent discount).

3.2 High dimensionality vs. data sparsity

Problem statement: In many use cases of DDN, we see *high-dimensional* relationships between session features and quality. The problem with such high dimensional structures is a classic curse of dimensionality. On one hand, as we combine many features, we will intrinsically have a *limited* number of sessions to infer the best decisions. On the other hand, as quality measurement could be different across similar sessions

	US sessions Sept 2012 [11]	% cities with 10 or less sessions/minute	% sessions in these cities
Netflix	382M	99%	76.6%
Hulu	694M	98%	65.2%
YouTube	16B	83%	11.8%
YouTube (Illinois)	973M	90%	22.6%

Table 3: ***Data sparsity manifest in even large application providers. A large fraction (highlighted) of sessions has 10 or less neighboring sessions in the same city every minute.***

due to intrinsic noise [43], we will need as many sessions as possible to achieve enough statistical confidence. For instance, video quality is impacted by combinational effect; e.g., bad quality happens for a specific ISP-city-CDN combination [32]. Suppose there are 100 ISPs, 1000 cities, then probing 10 CDNs, each with only one session, requires one million sessions, which is more than what ESPN has per hour (418M/month [4]).

Table 3 shows how such data sparsity manifests in three large video providers, under even a simple model. We assume their sessions are evenly spread over time and from all US cities in proportion to the population of each city [10]. We see that most cities have even less than 10 sessions per minute, and the sessions from these cities amount to a significant fraction (highlighted column), which suggests that failing to handle this issue would render DDN ineffective for a large portion of users.

Limitations of strawmen: Prior work has attempted to bridge the gap between high dimensionality and data sparsity, but it is unclear whether these solutions or application-specific insights can be generalized. One approach (e.g., [32]) identifies enough similar sessions to predict the quality of one single decision by focusing on critical features, but this does not always provide enough sessions to explore a large decision space where *each* decision needs to be predicted with enough sessions. Another approach (e.g., [30]) seeks to reduce the decision space, which may not be generalizable to where data sparsity results from skewed session distributions and high dimensional models (e.g., video streaming).

Opportunity: Contextual multi-armed bandits. We see an opportunity of casting DDN control logic as a *contextual multi-armed bandits* (CMAB) problem in machine learning [41]. CMAB techniques face a very similar setting to DDN; it needs to balance exploration and exploitation under sparse data and potentially complex reward function, which in our case, is the hidden function that determines application quality of a pair of session and decision (or "context").

However, CMAB techniques are no panacea, especially in a networking context. They often make too strict assumptions about the reward function; e.g., similar contexts always yield similar rewards [53], but in network applications, two sessions, who match on all feature except one, can still have very different network performance. They also assume that the reward function is fixed, but in network applications, sending too many sessions to the same resource can cause overload and congestion, thus altering the reward function.

We conjecture that CMAB techniques can be adapted with insights from prior work of DDN and standard networking techniques. For instance, instead of using generic similarity metrics, DDN can define similarity between video sessions by whether they match on critical features [32].

3.3 Scale, responsiveness and freshness

Problem statement: DDN raises challenges on the algorithmic front, as well as the architectural front. Even with a desirable CMAB algorithm, we still need an architecture for DDN controller that is *scalable* (i.e., scale to tens of millions of concurrent sessions per second [11]), *responsive* (i.e., respond to every client request within at most tens of milliseconds [24]), and able to make decisions based on *fresh* data. The required data freshness depends on how long a client can tolerate a suboptimal decision before DDN reacts, which, depending on application, is typically on the scale of seconds (e.g., live video has a buffer of 5-10 seconds).

Limitations of strawmen: From a pragmatic view, since application providers often store client provided data in widely distributed front-end clusters [45], a pure centralized controller (e.g., [16]) that gathers data to a central cluster is not favorable for DDN in most cases. A split control architecture (e.g., [24]) is the most promising strawman. It essentially trades data freshness for scalability and responsiveness, assuming that only single-session view has to be fresh while the multi-session view can tolerate several minutes of staleness. It is a reasonable tradeoff under stable workload where best decisions do not change so often, but during sudden events such as flash crowd, it will (temporarily) fall back to traditional local adaptation, at exactly the time when multi-session view is needed the most to spot the optimal decisions and maintain desirable quality and stability. Finally, recent work on distributed analytics studies a closely related problem, but it assumes either that data can be efficiently gathered to where the analytics happens [46] or that analytics results per cluster can be efficiently collected during the analytics [45]. Both assumptions are not (at least obviously) plausible, given millions of updates and requests that DDN needs to handle at scale. In short, we see a CAP-like conjecture that it is hard for DDN to simultaneously achieve scalability, responsiveness and data freshness. Thus, a tradeoff is needed, but what is a "good" tradeoff?

Opportunity: Regressograms. The key to strike a good tradeoff is the *scale of aggregation* in which DDN control loop must be real-time. Our insight, built on the machine learning concept of *regressogram* [49], is that the decision-making of a session depends only on a small group of "similar" sessions (e.g., the best server of a session can be infered by just looking at "neighboring" sessions who use same WAN path to each server), and each group is much smaller than the global view but large enough to avoid data sparsity. Now, if all operations (data collection, storage, and control logic) for a group are *localized* to one front-end cluster close to the sessions in the group, we will simultaneously achieve scalability, responsiveness, and limited yet *sufficient* data freshness.

The idea of regressogram can be implemented on top of existing systems [24] with two changes: (a) a learning algorithm run by the backend cluster to periodically partition sessions into groups on a relatively coarse timescale; and (b) a redirection service that

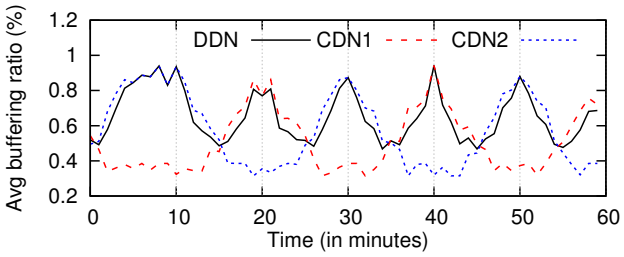

Fig. 4: ***Unstable control due to feedback delay. DDN observes degraded quality only after the CDN is overloaded (at each gray line), and oscillates between two CDNs.***

forwards the updates and requests of the same group to the same frontend cluster. The resulting architecture is similar to Social Hash [52] used in social networks to optimize hit rate of data retrieval. More broadly, we see a natural *synergy* between the algorithmic and architectural aspects of DDN, and the potential of a joint design for DDN controller.

3.4 Control stability

Problem statement: As in other time-delayed control systems [48], *feedback delay* will cause unstable behaviors in DDN, which negatively impact quality. Figure 4 uses trace-driven simulation to show how feedback delay coupled with load effect leads to oscillations. We consider two CDNs whose quality is extrapolated by real-world dataset and will degrade when overloaded [40]. Suppose DDN initially assigns most sessions to CDN1 and, due to feedback delay, DDN will react only after CDN1 is overloaded, and then switch sessions to CDN2, which again will be overloaded before DDN reacts, causing self-inflicted flash crowds and oscillations. Note that simple tricks like setting cap per CDN does not help, because DDN cannot identify if quality degradation is caused by its own decisions or not. Many causes can trigger this pathological phenomenon; e.g., when sessions arrive as flash crowd, and when multiple DDN instances run by different applications [31] or network layers [27, 33] simultaneously shift traffic between resources.

Though stability has been studied in many traditional network systems (e.g., [21, 47]), control stability has rarely been systematically studied in prior work of DDN.

Opportunity: "Benevolent" feedback delay. While the feedback delay in DDN is inevitable (e.g., to reliably measure packet loss rate at least takes several seconds), it is still possible to achieve stability despite of the feedback delay. One of the findings in control theory is that introducing positive random delay could sometimes *benefit* stability [12], which has found its early applications in network settings (e.g., [33]). We postulate that similar techniques can also help to stabilize DDN. Finally, to react to flash crowd in real time, DDN can "bypass" the quality measurement delay by simply monitoring the global workload (e.g., how many sessions are from each city) and available resources (e.g., how many sessions a CDN can handle).

4 Discussion

We end the paper by generalizing DDN design space, and revisiting epidemic problems in other DDN-like systems, such as single point of failure and data-driven bias.

4.1 Generalizing DDN design space

The earlier examples of DDN in §2.2 follow a specific deployment model: a logically centralized architecture within a single application provider (i.e., the data source and control system are in the same administrative entity). Now, we broaden the design space along two dimensions.

Degree of federation: Besides the single-provider model that we have seen, DDN can also be applied to a federated environment, where the DDN controller consolidates quality of sessions from different providers of the same application (e.g., small video sites use quality of YouTube sessions) or even of different applications or services (e.g., video providers informed by a CDN's internal server selection strategy [31]). Though DDN might benefit from having more data sources, it is not always feasible for application providers to share data with necessary fidelity or freshness. An exception, however, is where one company owns multiple applications and can effectively build a federated DDN within its borders [39]; e.g., Microsoft can use Bing query latency to infer WAN performance and optimize relay selection for Skype.

Degree of centralization: DDN controller can take a spectrum of degrees of centralization, spanning from physically centralized (i.e., in a single backend cluster), logically centralized (i.e., in globally distributed clusters with control logic driven by a potentially global view [24]), partially decentralized (i.e., in globally distributed clusters with control logic driven by a partial view of a subset of sessions, such as per edge AS [51]), as well as purely decentralized (e.g., each client runs a local control logic that can be updated based on a multi-session view on a coarse timescale [54]).

The challenges and opportunities outlined in §3 also manifest themselves in these design points to various degrees. What-if analysis in federated DDN is still challenging, but with more insights from other providers, the DR estimator might have fewer hidden features and less modeling bias. With more visibility and controllability, centralized and federated designs of DDN have less data sparsity and are relatively easier to be stabilized [40], but it will be more challenging for them to scale up and use fresh data. In addition, the control logic of federated DDN, like in any federated systems, must tolerate the uncertainties such as noise and error in the data shared by other providers [31, 15].

4.2 Open issues

Fault tolerance: In a logically centralized architecture, the DDN controller becomes a single point of failure. First, it is plausible that clients can (possibly unintentionally) launch a DDoS attack on the DDN backend by flooding control requests or quality updates. To address it, we can borrow from techniques of other centralized control platforms (e.g., SDN [34]). Second, a client may loose connection with the DDN controller due to firewall. Note that putting the controller in a highly available cloud will not address this issue. A promising alternative is the DDN client-side instrumentation automatically falls back to the control logic built into the client-side application (e.g., local bitrate adaptation of video players [24]) to achieve graceful quality degradation.

Algorithmic bias: Like any data-driven decision-making systems, DDN may suffer from algorithmic bias. To optimize the overall quality, DDN may use a small ISP, whose

sessions have small room for improvement, to explore suboptimal decisions, while allowing Comcast clients to exploit the optimal decisions, because they have greater room for improvement or simply because they have paid, which leads to a *reverse* network neutrality violation, where application providers discriminate ISPs! Preventing such bias is hard as data-driven algorithms are often "black box". These pitfalls make one wonder whether DDN will systematically hurt part of users. Fortunately, security and machine learning communities have started to shed light on these issues (e.g., [17, 55]), which could inspire ideas to prevent bias in DDN.

Acknowledgments

This research is supported in part by NSF award CNS-1345305 and NSF CISE Expeditions Award CCF-1139158, DOE Award SN10040 DE-SC0012463, and DARPA XData Award FA8750-12-2-0331, and gifts from Amazon Web Services, Google, IBM, SAP, The Thomas and Stacey Siebel Foundation, Adatao, Adobe, Apple Inc., Blue Goji, Bosch, Cisco, Cray, Cloudera, Ericsson, Facebook, Fujitsu, Guavus, HP, Huawei, Intel, Microsoft, Pivotal, Samsung, Schlumberger, Splunk, State Farm, Virdata and VMware. Junchen Jiang was supported in part by Juniper Networks Fellowship.

References

1. ACM SIGCOMM Workshop on QoE-based Analysis and Management of Data Communication Networks (Internet-QoE 2016). http://conferences.sigcomm.org/sigcomm/2016/qoe.php.
2. Bringing Data-Driven SDN to the Network Edge. https://www.sdxcentral.com/articles/contributed/network-edge-bringing-data-driven-sdn-to-the-network-edge-nick-kephart/2015/03/.
3. Emulab. https://www.emulab.net/.
4. ESPN, Inc. Fact Sheet. http://espnmediazone.com/us/espn-inc-fact-sheet/.
5. Periscope. https://www.periscope.tv/.
6. Spark. http://spark.incubator.apache.org/.
7. Technical note on the cfa algorithm. https://www.cs.cmu.edu/dda_technote.pdf.
8. The Data-Driven Approach to Network Management: Innovation Delivered. http://www.research.att.com/articles/featured_stories/2010_05/201005_networkmain2_article.html.
9. Twitch.tv. https://www.twitch.tv/.
10. US Census: City and Town Totals. http://www.census.gov/popest/data/cities/totals/2015/files/SUB-EST2015_ALL.csv.
11. U.S. online video platforms in September 2012. http://www.statista.com/statistics/271607/video-platforms-in-the-us-by-number-of-video-streams/.
12. C. T. Abdallah, R. Byrne, J. Benites-Read, and P. Dorato. Delayed positive feedback can stabilize oscillatory systems. In *Proc. ACC(American control conference)*, 1993.
13. A. Agarwal, S. Bird, M. Cozowicz, L. Hoang, J. Langford, S. Lee, J. Li, D. Melamed, G. Oshri, O. Ribas, et al. A multiworld testing decision service. *arXiv preprint arXiv:1606.03966*, 2016.
14. F. Chen, C. Zhang, F. Wang, and J. Liu. Crowdsourced live streaming over the cloud. In *INFOCOM*, 2015.

15. D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski. A knowledge plane for the internet. In *ACM SIGCOMM '03*.
16. D. Crankshaw, P. Bailis, J. E. Gonzalez, H. Li, Z. Zhang, M. J. Franklin, A. Ghodsi, and M. I. Jordan. The missing piece in complex analytics: Low latency, scalable model management and serving with velox. In *Conference on Innovative Data Systems Research (CIDR)*, 2015.
17. A. Datta, S. Sen, and Y. Zick. Algorithmic transparency via quantitative input influence. In *Proceedings of 37th IEEE Symposium on Security and Privacy*, 2016.
18. M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira. Pcc: Re-architecting congestion control for consistent high performance. In *Proc. NSDI*, 2015.
19. M. Dudík, J. Langford, and L. Li. Doubly robust policy evaluation and learning. In *Proceedings of International Conference on Machine Learning*, 2011.
20. N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin. An argument for increasing tcp's initial congestion window. *ACM SIGCOMM CCR*, 2010.
21. S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on networking*, 1(4):397–413, 1993.
22. S. Floyd and V. Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking (ToN)*, 2001.
23. A. Ganjam, V. Sekar, and H. Zhang. In-situ quality of experience monitoring: The case for prioritizing coverage over fidelity.
24. A. Ganjam, F. Siddiqi, J. Zhan, I. Stoica, J. Jiang, V. Sekar, and H. Zhang. C3: Internet-scale control plane for video quality optimization. In *NSDI*. USENIX, 2015.
25. A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *Intelligent Systems, IEEE*, 24(2):8–12, 2009.
26. O. Haq and F. R. Dogar. Leveraging the Power of Cloud for Reliable Wide Area Communication. In *ACM Workshop on Hot Topics in Networks*, 2015.
27. T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard. In *Proc. SIGCOMM IMC*, 2012.
28. V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM Computer Communication Review*, volume 18, pages 314–329. ACM, 1988.
29. V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *Proc. CoNext*, 2009.
30. J. Jiang, R. Das, G. Anathanarayanan, P. Chou, V. Padmanabhan, V. Sekar, E. Dominique, M. Goliszewski, D. Kukoleca, R. Vafin, and H. Zhang. Via: Improving Internet Telephony Call Quality Using Predictive Relay Selection. In *to appear in Proc. SIGCOMM*, 2016.
31. J. Jiang, X. Liu, V. Sekar, I. Stoica, and H. Zhang. Eona: Experience-oriented network architecture. In *ACM HotNets*, 2014.
32. J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang. CFA: a practical prediction system for video QoE optimization. In *Proc. NSDI*, 2016.
33. J. Jiang, V. Sekar, and H. Zhang. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Streaming with Festive . In *ACM CoNEXT 2012*.
34. R. Kandoi and M. Antikainen. Denial-of-service attacks in openflow sdn networks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1322–1326. IEEE, 2015.
35. S. Krishnan and R. Sitaraman. Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs. 2012.
36. S. S. Krishnan and R. K. Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. *IEEE/ACM Transactions on Networking*, 21(6):2001–2014, 2013.
37. A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, et al. BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing. In *Proc. SIGCOMM*, 2015.

38. L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.
39. H. H. Liu, R. Viswanathan, M. Calder, A. Akella, R. Mahajan, J. Padhye, and M. Zhang. Efficiently delivering online services over integrated infrastructure. In *Proc. NSDI)*, 2016.
40. X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang. A case for a coordinated internet video control plane. In *ACM SIGCOMM*, pages 359–370. ACM, 2012.
41. T. Lu, D. Pál, and M. Pál. Contextual multi-armed bandits. In *AISTATS*, pages 485–492, 2010.
42. H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iplane: An information plane for distributed services. In *USENIX OSDI '06*.
43. C. Pelsser, L. Cittadini, S. Vissicchio, and R. Bush. From paris to tokyo: On the suitability of ping to measure latency. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 427–432. ACM, 2013.
44. D. Precup, R. S. Sutton, and S. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.
45. Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica. Low latency geo-distributed data analytics. In *Proc. SIGCOMM*, 2015.
46. A. Rabkin, M. Arye, S. Sen, V. S. Pai, and M. J. Freedman. Aggregation and degradation in jetstream: Streaming analytics in the wide area. In *Proc. NSDI*, 2014.
47. J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. Bgp routing stability of popular destinations. In *Proc. SIGCOMM IMW*, 2002.
48. J.-P. Richard. Time-delay systems: an overview of some recent advances and open problems. *automatica*, 39(10):1667–1694, 2003.
49. P. Rigollet and A. Zeevi. Nonparametric bandits with covariates. In *Proc. Conference on Learning Theory*, 2010.
50. J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)*, 2(4):277–288, 1984.
51. S. Seshan, M. Stemm, and R. H. Katz. Spand: Shared passive network performance discovery. In *USENIX Symposium on Internet Technologies and Systems*, pages 1–13, 1997.
52. A. Shalita, B. Karrer, I. Kabiljo, A. Sharma, A. Presta, A. Adcock, H. Kllapi, and M. Stumm. Social hash: an assignment framework for optimizing distributed systems operations on social networks. In *Proc. NSDI*, 2016.
53. A. Slivkins. Contextual bandits with similarity information. *Journal of Machine Learning Research*, 15(1):2533–2568, 2014.
54. Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli. CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction. In *to appear in Proc. SIGCOMM*, 2016.
55. A. Vellido, J. Martin-Guerroro, and P. Lisboa. Making machine learning models interpretable. In *Proceedings of the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN). Bruges, Belgium*, pages 163–172, 2012.
56. S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica. Ernest: efficient performance prediction for large-scale advanced analytics. In *Proc. NSDI*, 2016.
57. K. Winstein and H. Balakrishnan. Tcp ex machina: computer-generated congestion control. In *Proc. SIGCOMM*, 2013.