Data-Driven Networking: Harnessing the "Unreasonable Effectiveness of Data" in Network Design

Junchen Jiang[*] **Vyas Sekar**[*] **Ion Stoica**[△†♮]
Hui Zhang[*†]

February 2016
CMU-CS-16-102

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[*]Carnegie Mellon University
[△]University of California, Berkeley
[†]Conviva Inc.
[♮]Databricks Inc.

This research was sponsored by the National Science Foundation under grant number CNS-1345305. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Data-Driven Networking, Network Architecture

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 of distributed 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

There is a revolution afoot in various aspects of computing driven by the ability to collect and extract "valuable insights" from large corpuses of data [2]. This "unreasonable effectiveness of data" [9] has sparked a radical rethink of how algorithms are designed—Rather than use human experts spending huge amounts of time and effort to craft sophisticated algorithms and complex models, we see *data-driven* models used that yield dramatically better results, and often with much simpler algorithms. Furthermore, this has opened new vistas to address grand challenge problems (e.g., NLP, speech, translation) that were previously considered unattainable.

We are seeing early evidence of this paradigm shift starting to reach the shores of networking motivated by two key technology trends. First, we observe a growing *decision space* of potential control decisions to optimize the performance of various network applications. For instance, recent work on C3 [7] shows how a logically centralized video control plane can take advantage of global visibility of network conditions to optimally tune "knobs" at the application layer w.r.t. CDN and bitrate selection. Second, we see increasing *heterogeneity* in operating conditions and application requirements, each requiring different control logic and parameters. For instance, recent work on Remy [26], shows how a machine-derived TCP could be tweaked to work better in different operating conditions.

This paper is a call-for-arms for the networking community to recognize and embrace this emerging *data-driven networking* (DDN) paradigm. Figure 1 conceptualizes the DDN paradigm and how it differs from traditional network design paradigms. In contrast to today's network designs that use only local data of a single flow with handwritten control logics, DDN suggests using *more data* from multiple flows and other sources, in order to make optimal control decisions.

(a) Non-DDN control loop (b) DDN-enabled control loop

Figure 1: Comparing control loops of a protocol when DDN is applied or not.

Early instantiations of this emerging DDN paradigm have already demonstrated significant potential benefits. We believe that by *systematizing* this paradigm as a broader community, we can unleash the unharnessed potential of DDN. Specifically, by creating a unified framework and broader community effort to think about DDN designs, we can uncover hitherto unexplored opportunities for DDN-inspired redesign of network applications and network protocols. Furthermore, we believe that this can shed light on key challenges in realizing the potential for DDN and allow us to extract a general set of reusable design principles and solutions to avoid the problem of each effort reinventing the wheel or worse ignoring some potential pitfalls.

In this respect, this paper makes four key contributions:

- First, we highlight illustrative examples that use existing instantiations of DDN to show that performance of today's protocols benefits from harnessing more data (§2).
- Second, we systematize these point solutions by developing a general framework for DDN. We use this framework to shed light on the existing instantiations of DDN and inspire new opportunites (§3).

- Third, we use this framework to highlight *common* challenges of applying DDN to any protocol, that would otherwise be difficult to formalize if seen from the narrow lens of a specific application. Moreover, we identify opportunities to develop general design guidelines to address these challenges (§4).
- Finally, as a case study, we show applying DDN can dramatically improve the design and performance of TCP, and use this case study to showcase potential solutions to the broader set of DDN challenges (§5).

2 Illustrative Examples

We begin with illustrative examples of existing efforts that highlight the early promise of the DDN paradigm.

2.1 Better CDN selection for video

The first example shows how quality measurements of many video sessions[1] can optimize CDN selection of individual video clients. Video players today have the flexibility of streaming content from one of multiple CDNs. However, with only information on a single flow, it is hard to infer the CDN of best performance without actually using each of them. Today, each video client is statically mapped to a CDN (at least the one a client starts with). Given performance of CDNs has a substantial diversity across geographical areas and temporal variability [19], there is a remarkable room of improvement by dynamically mapping a client to a CDN.

One data-driven approach [7] to dynamic CDN selection is by exploiting quality measurements of other (concurrent and history) video sessions. For instance, one can map a client to the CDN that has the best quality on similar video clients (e.g., those in the same AS and watching the same video content). In fact, prior work on C3 [7] showed a proof point of the viability and benefits of an Internet video control platform that aggregates video quality measurements from millions of video clients and uses them to inform CDN selection for future video clients. This operational system achieves better video quality than traditional local adaptation approaches; e.g., 50% reduction in the percentage of a session duration spent on re-buffering, while using similar bitrates.

2.2 Bitrate selection using TCP measurements

Ideally, a video player should pick the highest bitrate that is sustainable (i.e., below the throughput) to minimize buffering and ensure high user experience of video streaming. However, it is hard to accurately predict a session's performance, especially during the startup phase before a video session starts. Existing commercial approaches use either: (1) fixed bitrate for the duration of a video, which is intentionally set low to minimize buffering due to bandwidth fluctuation, or (2) even if they use adaptive bitrate streaming, start conservatively with a low bitrate and take a significant time to reach the optimal birate.[2]

[1] We use "flow" and "session" interchangeably, depending on convention (e.g., TCP flow, video session).

[2] Neflix `www.netflix.com/WiMovie/70136810?trkid=439131` takes roughly 25 seconds to switch from the initial bitrate of 560kbps to the highest sustainable bitrate (3Mbps).

	Prediction error	AvgBitrate	% bitrate > throughput
Today		2.5Mbps	11.8%
Data-driven (Decision tree)	3.4% (median)	13Mpbs	9%

Table 1: Improvement of data-driven initial bitrate compared to static initial bitrate.

It might, however, be feasible to predict a video session's throughput before it starts by using TCP throughput measured by other endpoints. Based on throughput predictions, one can pick the ideal (i.e., highest sustainable without buffering) bitrate [17]. Jiang et al., use a dataset of 9.9 million TCP throughput measurements taken from 6204 clients collected as part of the FCC's Measuring Broadband America platform [1], and build a Decision Tree model to capture the relation between throughput and session attributes (e.g., AS, connection medium, timestamp, etc) associated with each TCP flow. The result (Table 1) suggests that the ideal bitrate chosen based on throughput prediction can be $3\times$ higher than static bitrate and there are fewer sessions where the chosen bitrate exceeds the throughput.

This shows how data of different network layers enable accurate throughput prediction, which leads to better bitrate selection than using information of single video session.

2.3 Better `init_cwnd` by offline measurement

Setting a proper initial sending rate (`init_cwnd`) before a TCP session starts is inherently hard from the perspective of a single endpoint and thus `init_cwnd` has long been conservatively set to one packet. In turn, this makes many small files suffer from long completion time as they are RTT-bound. Realizing this, a proposal from Google made an empirical argument to increase `init_cwnd` to 10 packets based on large-scale measurements over a wide range of applications [6]. This work observed that average latency of HTTP responses improved by approximately 10% with the largest benefits being demonstrated in high RTT and high bandwidth-delay product (BDP) networks.

2.4 Rethinking TCP design

Today's TCP control logics use "magic" constants handed down from generations that are manually configured (e.g., the multiple and increment of AIMD). However, the optimal values of these constant should be a function of the network context (e.g., data center, satellite network, etc) in which TCP operates, implying significant room of improvement through setting optimal values on these constants. Prior work on Remy [26] showed that a machine designed TCP that uses offline simulation driven by prior knowledge of the network context (e.g., topology, degree of multiplexing) can learn the best values of some key constants and achieve significant performance improvement over today's manually picked values; e.g., a simulated 15 Mbps fixed-rate link with eight senders contending and an RTT of 150 ms, Remy control logic achieves 40+% throughput speed up and 20+% delay reduction over many specially engineered TCP variants.

Finally, we show an example of how TCP can be simplified by driving the control decisions

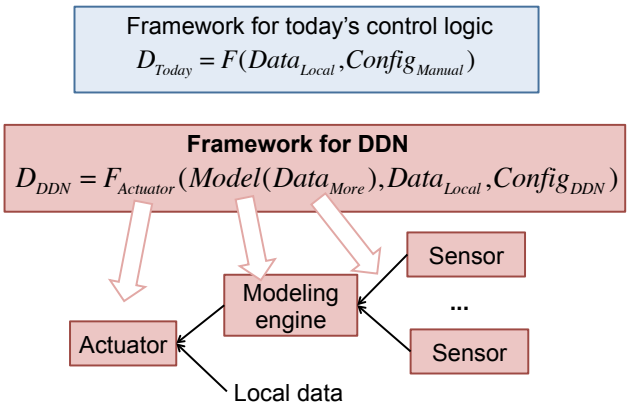

Figure 2: Framework for DDN.

directly by data on performance measurements. Today's TCP congestion control relies on various packet-level network events as signal of network conditions. To this end, TCP uses some logic to map predefined packet-level events to control responses on sending rate (`cwnd`) based on indirect inference on certain network conditions, which is often not accurate. In contrast, PCC [5] attempts to simplified TCP congestion control algorithm by using locally observed performance to directly drive the setting of `cwnd`. Specifically, it continously maintains a model between observed performance (e.g., throughput and latency) and `cwnd`. This approach greatly simplifies TCP congestion control, while at the same time, significantly improves TCP performanc; e.g., $10\times$ higher throughput of TCP CUBIC on global commercial Internet.

While Remy and PCC's designs were developed in closed world simulations, we argue that it is not a significant leap of faith to imagine this design process could have itself been a data-driven step run "in the wild" on real sessions.

3 Framework of DDN

In this section, we present an attempt to formalize the DDN paradigm and discuss a simple taxonomy of DDN instantiations based on when and how DDN can be used. We believe that having such a systematic framework will in turn highlight new opportunities for further unleashing the benefits DDN and also act as an anchor to highlight the key technical challenges in realizing the potential of DDN (§4).

3.1 What is DDN?

Definition of DDN: In a nutshell, DDN optimizes the performance of a single flow by leveraging *all* data available from other flows or other network elements. This is in contrast to the existing performance optimization techniques which typically use only data specific to the flow being optimized. Figure 2 illustrates the overall framework of DDN.

The capability of harnessing more data enables DDN to make better control decisions. For instance, C3 (§2.1) optimizes control decisions of CDN and bitrate selection by taking quality measurements from many clients, rather than one client, as input data to the control logic. In

addition, DDN enables data-driven setting on the configuration. For instance, `init_cwnd` (§2.3) is increased from one to ten packets not based on manual selection, but by empirical measurements of performance under different values of `init_cwnd`.

Workflow of DDN: The "control formula" of DDN in Figure 2 naturally suggests three logical modules in DDN:

- **Sensor:** First, DDN uses sensors to gather input data from different sources. For instance, the sensing layer of C3 measures quality on client-side players via instrumentation [3].
- **Modeling engine:** Second, DDN uses a modeling engine to identify spatial and/or temporal correlations among input data (collected by sensors), and build a aggregated model, which informs control decisions. For instance, C3 builds a prediction model of video quality based on real-time quality measurements of millions of clients.
- **Actuator:** Finally, DDN uses actuators to make control decisions for individual flows based on aggregated model produced by the modeling engine. For instance, actuators of C3 make control decisions w.r.t CDN/bitrate selection based on the quality prediction model and the local information of a specific video client.

This high-level view can now help us envision a broad design space of DDN instantiations; e.g., depending on the type of data, spatiotemporal granularity of data, and how/when the data-driven models are used as part of the actuation logic.

3.2 What is input data of DDN?

We classify the "sensor" data for a DDN instantiation along two axes: type (e.g., source of the data) and granularity.

Type: Input data can be collected from different devices, different protocols, and even different layers. For instance, to optimize control decisions of video streaming, one could leverage performance measurements of other video sessions on other devices (e.g., §2.1), cross protocols (i.e., web sessions), and cross-layer (e.g., bitrate selection based on TCP measurements in §2.2).

Besides performance information, there are other types of input data that can inform DDN to make optimal control decisions. For instance, path or link-level information from ISPs [15] or other sources (e.g., [20]) allows DDN to accurately identify similar TCP flows that traverse the same bottleneck link. Moreover, accurate simulation on network state transition (§2.4) also provide insights on how control logics perform in a certain setup.

Granularity: Input data may also differ in spatial and temporal granularity. It can be "local" (e.g., PCC §2.4), "global" (e.g., C3 §2.1), or somewhere in between (e.g., a small collection of hosts [25]). Furthermore, it can be real-time (within seconds), near real-time (with delay of minutes to inform control decisions with more persistence) or offline (e.g, §2.3).

3.3 How does DDN improve a protocol?

Next, we focus on possible instantiations of DDN-inspired novel actuation mechanisms. We classify them along two axes: *how* and *when* DDN performs the "actuation" step.

"How"–Decisions or configurations: DDN can be integrated in a protocol in different ways. The clean-slate (and radical) approach is that DDN uses input data to directly drive control decisions.

However, this requires a wholesale refactoring over today's control logic (which is not designed to use more input data), and is thus not ideal for wide deployment, especially for applications that offer limited interfaces (e.g., video streaming protocols are mostly proprietary). Alternatively, one may use data to drive the selection of configuration parameters (e.g., `init_cwnd`) or control logic based on the network context.

"When"–Online or offline: Both control logics and configuration of control logics can happen either online or offline. Ideally, one may want to continously tune control decisions to capture dynamics of network conditions and traffic pattern. Nevertheless, some control decisions are infeasible to be changed in real time and therefore are better to be set offline. For instance, the list of bitrate levels in which a video is encoded is critical to video quality. But they cannot be easily reconfigured as it requires re-encoding videos and disseminating them to CDN edge servers.

3.4 What is the application of DDN paradigm?

At a high level, DDN paradigm should be applicable to any network protocols whose performance depends on control decisions. In particular, DDN has larger potentials when optimal decisions depend on locally invisible elements, for which DDN offers more visibility with more input data. This applies to most endpoint protocols which continously make control decisions to cope with the varying network conditions. DDN also benefits protocols of internal network elements (e.g., routing) to make better control decisions in response to dynamic traffic patterns. In particular, we present some concrete usecases of DDN by extending existing point instantiations and shedding light on new applications.

Unexplored opportunities for TCP: Based on the framework of DDN, we identify two unexplored opportunities for TCP. First, rather than picking a one-size-fits-all `init_cwnd` based on empirical measurements (§2.3), it can be extended in DDN framework to dynamically customize `init_cwnd` for different TCP flows based on real-time data. Given the considerable spatial and temporal variability of bottleneck bandwidth, it would unleash significant improvement by dynamically use the largest `init_cwnd` that would have no loss. Second, rather than building a model between `cwnd` and performance with local information (§2.4), it can be extended in DDN framework to combine the observed performance from multiple TCP flows and thus identify the optimal `cwnd` more efficiently.

Skype: To ensure high quality (e.g., low loss rate, latency and jitter), Skype tunes "control knobs", such as the relay paths (i.e., supernodes that connect two clients) in order to avoid links with congestion or high loss. Today, such selection is largely agnostic to real-time network conditions. With access to multiple flows, it is feasible to infer current network conditions and make better decisions based on quality of concurrent and historical sessions. For instance, if Skype calls between two IP prefixes show better performance on one relay path than those on another path, the latter relay path is likely to be congested, and we should re-route the traffic to avoid the congestion. Alternatively, knowing the congestion can inform the decision to place an additional supernode, if the congestion cannot be circumvented via network routing. All these would be infeasible without the visibility of many application sessions.

Routing: Today's BGP prefers shorter AS-paths as an indirect way to improve end-to-end performance. However, the shortest AS-level path is not necessarily the path that offers the best

end-to-end performance [24]; e.g., traversing a large AS that have many hops for intra-domain routing will be slower and more likely to be congested than going through multiple small ASes. DDN can alleviate this problem by selecting path based on real-time measurements of end-to-end performance. They can be measured by edge ASes or sharing information across ISPs and endpoints (similar to EONA [15]). Similarly, we see a natural synergy between DDN and emerging network management paradigms such as SDN [21] and SDX [8] that use a logically centralized network-wide view. To date, however, SDN deployments still use traditional decision (e.g., traffic engineering algorithms [14, 11]). We believe that going forward SDN too can benefit by making the control logic DDN-like.

4 Challenges

The unified DDN framework allows us to elaborate *common* challenges faced by applying DDN on any protocol. Furthermore, it sheds light on opportunities, general-purpose design guidelines and even reusable solutions to address the challenges. In essence, the key challenges of DDN lie in *where* and *how* the three modules of DDN– sensors, modeling engine and actuators (Figure 2) – should be implemented.

4.1 Sensors: Scalable sharing platform

The first challenge concerns where and how sensors share measured data efficiently. The nature of networks means sensors must be implemented in a distributed manner. Once data are measured (which in itself is a challenge yet beyond the scope of this paper), it is challenging to share them efficiently to inform the decisions of as many flows as possible, especially in wide-area networks. Hence, a practical solution must strike a balance between more visibility and more freshness of data. On one hand, decentralized sharing mechanism may provide fresh information, but it is not scalable to propagate it among many endpoints (i.e., less visibility). On the other hand, a centralized entity can collect data of a global view, but it is hard to get real-time information given the inevitable latency to endpoints.

Opportunity: Tradeoff between global visibility and fresh information. Many control decisions can trade global visibility for information freshness or vise versa because they do not require global visibility and freshest information at the same time. The underlying rationale is that there is lower dynamics (hence more tolerance on stale data) on a global scale than on a local scale. For instance, it has been shown [7] that, although finding the best CDN requires a global view (video sessions using different CDNs), the best CDNs can be inferred from slightly stale data (e.g., with several minutes delay) and cached because they tend to persist on a timescale of minutes. In contrast, control decisions that require real-time data often do not need data from global scope. For instance, video bitrate adaptation, under sudden changes of bandwidth, needs real-time throughput measurements, but history throughput of a single flow is usually sufficient.

Design guideline: Two-tier design. Given the tradeoff between global visibility and fresh information, a practical design for DDN sensor would be to collect data in different timescales: data that require more global visibility but lower freshness can be collected by a centralized entity which

operates on a coarse-grained timescale (e.g., of minutes), while data that require more freshness but less visibility can be collected by decentralized entities which operate on a fine-grained timescale (e.g., milliseconds or seconds) and closer to the network elements being optimized.

4.2 Modeling engine: Effective aggregation of data

The second challenge regards how DDN modeling engine performs effective analytics and spatial/temporal aggregation over data from different sources. This is challenging for a number of reasons. First, it must take into account the inherent "noise" embedded in the data. For instance, video quality measurements on different CDN can be biased by unbalanced distribution of population (e.g., most video sessions use certain CDN due to policy) or hidden factors (e.g., CDN selection is based on unknown contract between ISPs and CDNs). The negative impact of unbalanced data is a common caveat when applying statistics techniques [4]. Second, the data may have complex and dynamic underlying structures. For instance, TCP flows could be bottlenecked by different links, which vary spatially and temporally. Such structures must be accurately identified, or the control decisions will be made with irrelevant information and become random or worse off. Third, because control decisions are often needed immediately based on fresh information, the modeling engine must be scalable to run (potentially complex) analytics very quickly.

Opportunity: Critical and persistent correlation[3] between flows. Though correlations among data could be complex and dynamic, it has been shown that key factors that impact performance are persistent and identifiable. For instance, measurement study on video quality issues [16] has shown that poor quality can be attributed to one or two key attributes (e.g., CDN or content availability), and the impact of these attributes tends to persist for a long time, which implies that DDN can learn such key correlations based on history data.

Design guideline: domain-specific learning of correlations To learn the correlation from data, we can borrow a rich literature in statistics and machine learning on learning data correlation by applying domain-specific insights; e.g., each flow can be viewed as a data point associated with domain-specific features that could potentially impact the performance (e.g., initial bitrate selection in §2.2).

4.3 Placing actuators: Minimal changes to today's protocols

The third challenge concerns where actuators should be implemented in order to minimize changes to existing protocols. A clean-slate DDN approach would replace today's control logic by an actuator that can utilize more data. This, however, may require a wholesale change on control logics of today's protocols, and there is a number of reasons to keep minimal changes on existing protocols. First, minimal changes preserve desirable properties of today's implementations (e.g., packet conservation of TCP [13]) and maximize compatibility with legacy implementations (e.g., TCP friendliness). Second, many application-level protocols are proprietary and offer limited interfaces (e.g., Standard video DASH protocols only allow third-party change on CDN and bitrate at the beginning of a session), so holistic changes may not be realistic or have wide deployment.

[3]Correlation is used to refer to both flows that have similar performance and the correlation between the flow attributes and its performance, which is an indirect indicator of correlation among flows

Opportunity: Exploring configuration parameters Previous research has shown a significant impact of configuration on performance, even the control logic remains unchanged. For instance, Remy (§2.4) parameterizes TCP and achieves remarkable performance improvement by only setting the configuration parameters.
Design guideline: Setting key constants dynamically. DDN can still achieve better performance with minimal changes by optimally setting key configurations in today's control logics. This avoids a wholesale change on the control logics.

4.4 Logic of actuators: Stable and optimal decision making

The last challenge concerns how actuators make control decisions. In reality, it is possible that DDN needs to collect information from the same endpoints that the actuator makes decision for (e.g., C3). However, this could create oscillation in the control loop and suboptimal control decisions. For instance, if many video clients switch to the most uncongested CDN simultaneously, it can create congestion on one CDN and cause the video clients to switch again. Even if a CDN is not congested by moving all clients to it, moving all clients to the same decision reduces the visibility to different decisions and may stuck in local optimal decision.
Opportunity: Access to many flows. To strike a balance between stability and optimality, DDN could leverage the control over many flows and direct flows over multiple choices. In general, the capability to explore multiple choices simultaneously can lead to more stable and efficient control (e.g., [10]). For instance, in order to find the best inter-domain AS-PATH, one should use a small fraction of traffic to explore the paths other than the optimal one [24], which offers visibility on other paths in case performance of current path degrades.
Design guideline: Tools of machine learning and control theory. In essence, this challenge can be framed as exploration-exploitation problem, which is a rich research topic in it own right (e.g., active learning and multi-bandit [18] problems). Yet, conventional techniques assume that the "ground truth" of each decision is independent to the decisions, which often does not hold for network protocols; e.g., CDN's performance does change under different loads [19]. So DDN actuators should also apply adaptive control to stabilize the control (e.g., [22]).

5 Case Study: DDN-TCP

This section instantiates the framework (§3), challenges and potential solutions (§4) of DDN in a specific application of TCP, called DDN-TCP. We present the design of sensor, modeling engine and actuator, and discuss open challenges.

5.1 DDN-TCP Sensors

Previous research has shown that TCP can benefit from data from different sources. Ideally, fine-grained network feedback, such as ECN [23], helps TCP make more informed decisions. In addition, sharing congestion states maintained by different flows can also lead to significant improvement [25]. Besides congestion states and packet-level eventsi, performance measurements (e.g., throughput or latency) of different `cwnd` also help TCP congestion control to find the optimal

cwnd [5]. Even offline simulation [26] driven by network statistics can be used as input data to inform better congestion control logics. In the following discussion, we assume that sensors collect performance measurements of many TCP flows from different endpoints (a design point that has not been discussed).

There are two open questions regarding sensors of DDN-TCP. First, in order to collect or share the data, we need standard formats that specify low-level details such as measurement methodology, units and precision, and so forth. We believe that some standard body (e.g., IETF) will precisely define these semantics. Second, because sharing data consumes precious bandwidth, DDN-TCP needs lightweight sharing mechanisms that have little bandwidth overhead and can work over unreliable connections.

5.2 DDN-TCP Modeling engine

After receiving performance measurements from sensors, the modeling engine identifies correlations between TCP flows. DDN-TCP learns the correlation among flows by a back-end cluster, which has the resource to run complex inference techniques (e.g., SVM) (§4.3). Though using a centralized modeling engine would add inherent delay to the identified flow correlation, this is tolerable due to the persistence of such correlation (§4.1); e.g., bottleneck links shared by correlated flows tend to have a long persistence [12].

There are two open questions regarding the modeling engine of DDN-TCP. First, aggregating data by a centralized cluster may cause privacy issues or who should run/own the platform of DDN-TCP. Second, TCP performance can be impact by some factors that are invisible to the modeling engine. To avoid such negative impact, the modeling engine should isolate the TCP flows whose performance does not show strong correlation with any available information.

5.3 DDN-TCP Actuators

Finally, for the purpose of minimimal changes to the existing protocols (§4.3) and fault tolerant, each endpoint should still run today's congestion control. DDN-TCP actuators should be operated by separate machines, and set the key configuration (e.g., init_cwnd) of local TCP and send congestion control decision of cwnd when the actuator decides to change it. Because actuators need to be responsive to packet losses, latency between actuators and endpoints must be lower than RTT of the TCP flow being optimized. Therefore, unlike the modeling engine, DDN-TCP uses geographically distributed front-end actuators to minimize latency to different endpoints. We design the workflow of DDN-TCP as an extenion of PCC (§2.4). The modeling engine identifies correlated flows (i.e., those sharing the same bottleneck), and each front-end actuator compares the performance of different cwnd over these correlated flows simultaneously. This could yield similar benefits as PCC and have more efficient convergence to optimal cwnd.

Figure 3 shows some early results of DDN-TCP about how much flow completion time can be reduced by setting init_cwnd dynamically rather than statically. We simulated a bottleneck link (bandwidth b, queue length of q and delay d) shared by n competing flows, each sending file of size f. The figure shows the distribution of reduction of flow completion time compared with using static init_cwnd of 1 or 10 packets. The distribution is over all combinations of $b =$

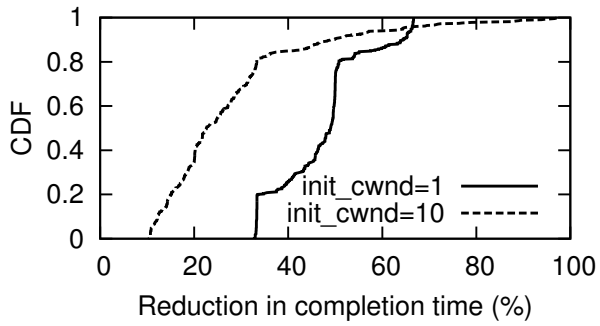

Figure 3: Early results of DDN-TCP

$\{1, 10, 100\}$Mbps, $q = \{10, 100, 1K, 10K\}$pkts, $d = \{1, 10, 100, 1000\}$ms, $n = \{1, 2, 4, 8, 16\}$, $f = \{1.5, 6, 24, 96, 384\}$Mb. It shows a median reduction of 20% or 50% compared to static `init_cwnd`, and up to 97% reduction when q is small (where `init_cwnd` =10 is too aggressive and causes severe packet loss).

There are two open questions regarding actuators of DDN-TCP. First, actuators must have fast reaction to keep packet conservation [13] and avoid congestion collapse, but the delay between endpoints and front-end actuators could cause slow reaction to network congestion. Second, in a high-level, DDN-TCP actuator need to be "fair" to other flows, though the definition of such fairness is beyond the scope of this paper.

References

[1] 2014 measuring broadband america report technical appendix. `http://data.fcc.gov/download/measuring-broadband-america/2014/Technical-Appendix-fixed-2014.pdf`.

[2] Daniel Crankshaw, Peter Bailis, Joseph E Gonzalez, Haoyuan Li, Zhao Zhang, Michael J Franklin, Ali Ghodsi, and Michael I Jordan. The missing piece in complex analytics: Low latency, scalable model management and serving with velox. 2015.

[3] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Antony Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. In *Proc. SIGCOMM*, 2011.

[4] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[5] Mo Dong, Qingxi Li, Doron Zarchy, P. Brighten Godfrey, and Michael Schapira. Pcc: Re-architecting congestion control for consistent high performance. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 395–408, Oakland, CA, 2015. USENIX Association.

[6] Nandita Dukkipati, Tiziana Refice, Yuchung Cheng, Jerry Chu, Tom Herbert, Amit Agarwal, Arvind Jain, and Natalia Sutin. An argument for increasing tcp's initial congestion window. *Computer Communication Review*, 40(3):26–33, 2010.

[7] Aditya Ganjam, Faisal Siddiqi, Jibin Zhan, Ion Stoica, Junchen Jiang, Vyas Sekar, and Hui Zhang. C3: Internet-scale control plane for video quality optimization. In *To appear in NSDI*. USENIX, 2015.

[8] Arpit Gupta, Muhammad Shahbaz, Laurent Vanbever, Hyojoon Kim, Russ Clark, Nick Feamster, Jennifer Rexford, and Scott Shenker. Sdx: A software defined internet exchange. 2014.

[9] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *Intelligent Systems, IEEE*, 24(2):8–12, 2009.

[10] Eshcar Hillel, Zohar S Karnin, Tomer Koren, Ronny Lempel, and Oren Somekh. Distributed exploration in multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 854–862, 2013.

[11] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. In *ACM SIGCOMM 2013*.

[12] Ningning Hu, Li Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. A measurement study of internet bottlenecks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1689–1700. IEEE, 2005.

[13] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM Computer Communication Review*, volume 18, pages 314–329. ACM, 1988.

[14] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a globally-deployed software defined wan. In *ACM SIGCOMM 2013*.

[15] Junchen Jiang, Xi Liu, Vyas Sekar, Ion Stoica, and Hui Zhang. Eona: Experience-oriented network architecture. In *ACM HotNets*, 2014.

[16] Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. Shedding light on the structure of internet video quality problems in the wild. In *CoNEXT*. ACM, 2013.

[17] Junchen Jiang, Vyas Sekar, and Yi Sun. Dda: Cross-session throughput prediction with applications to video bitrate selection. *arXiv preprint arXiv:1505.02056*, 2015.

[18] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

[19] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A case for a coordinated internet video control plane. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 359–370. ACM, 2012.

[20] Harsha V Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iplane: An information plane for distributed services. In *USENIX OSDI '06*.

[21] Nick McKeown. Software-defined networking. *INFOCOM keynote talk*, 17(2):30–32, 2009.

[22] Andrew Y Ng and H Jin Kim. Stable adaptive control with online learning. In *NIPS*, 2004.

[23] Kadangode Ramakrishnan and Sally Floyd. A proposal to add explicit congestion notification (ecn) to ip. Technical report, RFC 2481, January, 1999.

[24] Michael Schapira, Yaping Zhu, and Jennifer Rexford. Putting bgp on the right path: A case for next-hop routing. In *ACM HotNets '10*.

[25] Srinivasan Seshan, Mark Stemm, and Randy H Katz. Spand: Shared passive network performance discovery. In *USENIX Symposium on Internet Technologies and Systems*, pages 135–146, 1997.

[26] Keith Winstein and Hari Balakrishnan. Tcp ex machina: Computer-generated congestion control. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 123–134. ACM, 2013.