Towards Minimal-Delay Deadline-Driven Data Center TCP

Li Chen
CSE, HKUST
lchenad@cse.ust.hk

Shuihai Hu
CSE, HKUST
shuaa@cse.ust.hk

Kai Chen
CSE, HKUST
kaichen@cse.ust.hk

Haitao Wu
Microsoft Research Asia
hwu@microsoft.com

Danny H. K. Tsang
ECE, HKUST
eetsang@ece.ust.hk

ABSTRACT

This paper presents MCP, a novel distributed and reactive transport protocol for data center networks (DCNs) to achieve minimal per-packet delay while providing guaranteed transmission rates to meet flow deadlines. To design MCP, we first formulate a stochastic packet delay minimization problem with constraints on deadline completion and network stability. By solving this problem, we derive an optimal congestion window update function which establishes the theoretical foundation for MCP. To be incrementally deployable with existing switch hardware, MCP leverages functionality available on commodity switch, i.e., ECN, to approximate the optimal window update function. Our preliminary results show that MCP holds great promise in terms of deadline miss rate and goodput.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design, Performance

Keywords

Data Center Networks, TCP, Deadline, Stochastic optimization

1. INTRODUCTION

Cloud datacenter applications such as web search, retail, advertising, and recommendation systems, etc., generate a diverse mix of short and long flows that carry widely varying deadlines [1, 6, 10, 11] due to their soft-real time nature. Flows that fail to finish within their deadlines will not contribute to the application throughput and will be excluded from the net results. This severely wastes network bandwidth, affects user-perceived experience, and thus causes provider revenue loss [11].

However, today's datacenter transport protocols such as TCP, given their Internet origins, are oblivious to such flow deadlines which have already caused problems. For example, after investigating a couple of production DCNs, people have witnessed a substantial fraction (from 7% to over 25%) of flow deadlines are not met, significantly degrading application response quality and incurring operator revenue loss [11].

To address this issue, recent new incrementally-deployable designs like DCTCP [1] and D2TCP [10] have been proposed for DCNs. While DCTCP focuses on achieving high throughput for long flows and low latency for short ones, D2TCP further considers meeting deadlines as its primary objective. Despite generally improving packet latency by enforcing small queue with low ECN marking threshold [5], they lack a theoretical foundation to achieve minimal delay or to meet flow deadlines. Worse, as shown later, these schemes are fundamentally constrained because they cannot precisely estimate or are, by design, not able to provide the right rates for deadlines. For example, DCTCP exhibits poor performance when the required flow rates are higher than their fair-sharing rates to meet their deadlines. While D2TCP uses deadline information to modulate congestion window after congestion occurs, i.e., far-deadline flows backoff more and near-deadline ones backoff less, it is ineffective for long flows with deadlines. This is because in the beginning long flows behave similarly to DCTCP and always backoff for incoming flows, whereas the increase in rate in the later stage when their deadlines approach may already not be enough to catch up (Section 2).

While explicit rate control mechanisms like D3 [11] and PDQ [6] can potentially solve the above problems by scheduling flows and assigning rates according to their sizes and deadlines in a coordinated manner, they require non-trivial switch hardware modifications and are quite challenging to implement in practice. Most recently, pFabric [3] achieves near-optimal flow completion times (thus better deadline meeting rate) by the novel idea of decoupling flow scheduling from rate control. However, it is a clean-slate design that requires modifications on both network switches and end hosts.

In this paper, we present MCP, a novel distributed and re-

Hotnets '13, November 21–22, 2013, College Park, MD, USA.

active transport protocol for DCNs while still can provide the *right* transmission rates to meet flow deadlines and achieve minimal per-packet delay. When designing MCP, we explicitly set out our design goals of not modifying switch hardware and supporting incremental deployment. Thus MCP is along the line of TCP variants adapted for DCNs such as DCTCP and D2TCP, however, the key difference is that our MCP builds up on the theoretical foundation we established for packet delay minimization and flow deadline completion.

The key contributions of our work are as follows.

- **We establish a theoretical foundation for optimal distributed rate control.** We formulate a stochastic packet delay minimization problem with constraints on deadline completion and network stability. We then apply the Lyapunov optimization framework to transform this problem to a convex problem, so that an optimal congestion window update function can be derived from the optimal solution for the transformed convex problem. Our analysis confirms the stability and the optimality of the algorithm.
- **We design MCP, a practical near-optimal transport control protocol.** Guided by the theory, we design MCP a practical implementation of the derived optimal algorithm. The key goal of MCP is to leverage commodity switch available functionalities to approximate the optimal congestion window update function. Thus, we propose an approximation method to estimate the parameters in the optimal window update function purely based on the ECN feedback bits, so that no hardware modification is needed, and MCP can be incrementally deployed to existing DCNs.

2. A MOTIVATING EXAMPLE

We identify the fundamental limitation of both DCTCP and D2TCP using an illustrative example in Figure 1, which motivates our design of MCP.

With DCTCP [1], due to its fair-sharing nature, for N flows sharing a network link with capacity C and a switch buffer with ECN marking threshold K, the rate of each flow is upper-bounded by $\frac{C+K/RTT}{N}$. This is because when the buffer queue size reaches K, the switch starts to react by marking ECN packets. In the subsequent RTT, the sender will reduce its congestion window accordingly. Thus, the rate of DCTCP can be roughly described as the blue sawtooth in Figure 1. In this case, for a flow that requires more than this limited rate to meet its deadline, e.g., the red line, DCTCP is by no means able to provide the necessary rate to cater for the requirement.

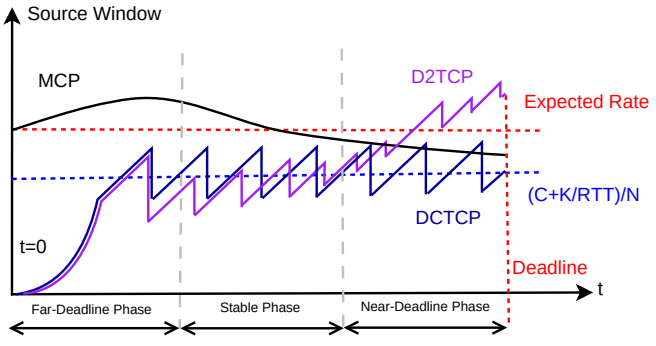

Figure 1: Motivating example

D2TCP [10] builds up on DCTCP and adds deadline awareness on top of it. It changes the congestion window update function to incorporate deadline information when congestion is detected: far-deadline flows backoff more, and near-deadline flows backoff less. Thus, the windows size of D2TCP can be roughly depicted as the purple sawtooth in Figure 1. In far-deadline phase, D2TCP backoffs more than DCTCP, and in stable phase, D2TCP operates very similarly to DCTCP, and have to give up bandwidth if new flows joins the network. It is evident that, while the D2TCP's deadline-aware backoff strategy certainly helps in some cases, it still cannot satisfy the requirement specified by the red line. In near-deadline phase, the increased rate is not enough to make up for the deficit in the previous phases. One key reason is that D2TCP uses deadline information in its backoff in near-deadline phase, which is already too late for it react to the stringent deadlines.

The key takeaway from the above analysis is two-fold. First, explicitly enforcing small ECN threshold helps to achieve low latency because packets always see small queues, however, it imposes a fundamental constraint on flow rate and is harmful for flows that require higher rates to meet their deadlines. To prevent this rate-limiting behavior, we need a better scheme to minimize the latency while not violating the deadlines. Second, in order to meet deadline, the deadline information should be respected throughout the lifetime of a flow instead of merely in the backoff near-deadline stage which might be too late to take effect. With flow size and deadline known at the source before the transmission, the expected rate can be determined and should be achieved throughout its lifetime.

This motivates our design of MCP. MCP is a protocol that fully utilizes ECN feedback to determine a right transmission rate for each flow. First, the rate should be high enough to satisfy the flow deadline requirement. Second, the rate should be as low as possible to ensure packets from short flows experience low latency. The behavior of MCP window update mechanism is shown in Figure 1, the deadline information impacts the flow throughout its lifetime. Sources adapt their transmission rates according to the required rates of flows to meet deadlines and the current network congestion.

Note that such desirable properties of MCP are achieved not by a heuristic design, but by the analysis built upon a theoretical foundation. We formulate a stochastic delay minimization problem with constraints on deadline completion, and we derive an optimal congestion window update function from its solution. The theoretical optimal window update function guides the design of MCP, a practical transport protocol that ensures the right transmission rates to meet flow deadlines, while achieving the minimal delay.

3. THEORETICAL FOUNDATION

3.1 System Model

Consider a DCN with L logical links, each with a capacity

of C_l bits per second (bps). In the network, the total number of active sessions is S. At time t, session s transmits exactly one flow at a rate of $x_s(t)$ bps, and the remaining data size is denoted as $M_s(t)$, and the remaining time till deadline $\delta_s(t)$. Like D2TCP [10] and D3 [11], we assume applications pass deadline information to the transport layer in the request to send data. Define $\gamma_s(t) = M_s(t)/\delta_s(t)$ as the expected rate for session s. We also do not consider the routing of the flow, and assume that the flow from session s will be routed through a fixed set of links $L(s)$. For link l, denote y_l as the aggregated input rate to link l, and $y_l = \sum_{s \in S(l)} x_s$, where the set of flows that pass through link l is denoted as $S(l)$.

3.1.1 Minimal Delay

Packet delay should be minimized, as the deadline completion for short and query flows is sensitive to it. As discussed in Section 2, both DCTCP and D2TCP use small ECN marking threshold to obtain low delay, but this method imposes rate limiting and may hurt the deadline completion of flows. On the other hand, TCP is inherently aggressive, as the shortsighted sources always expand for more bandwidth and may build up long queue consequently, increasing the latency. We believe this inherent aggressiveness should be counteracted by considering the long term average of network metrics. Instead of limiting the ECN threshold, we decide to use the long term average of per-packet delay as the minimization objective of our formulation.

Denote $d_l(y_l)$ the delay that a packet experienced on link l with load y_l. For session s, the average packet delay is $\sum_{l \in L(s)} d_l(y_l)$. The delay of link l, $d_l(y_l)$, is a function of y_l, the aggregated arrival rate at link l. $d_l(y_l)$ is a positive, convex and increasing function. We define the objective function as the long term average of the summation of per-packet delay of every source.

$$P_0(\mathbf{x}, \mathbf{y}) = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{s} \{ \sum_{l \in L(s)} d_l(y_l(t)) \} \tag{1}$$

3.1.2 Deadline Guarantee

In the our formulation, the long term average transmission rate is required to be larger than the expected rate. This constraint is an approximation to the realistic DCN traffic, where the flows cannot be infinitely long.

$$\lim_{t \to \infty} \frac{\sum_0^t (\gamma_s(t) - x_s(t))}{t} \leq 0, \forall s \tag{2}$$

By incorporating this constraint, we are essentially guaranteeing that, for every flow that requires γ_s, the transmission rate x_s is on average larger than γ_s.

3.1.3 Network Stability

Denote the instantaneous queue length at link l as $Q_l(t)$. The rate stability condition is therefore: $\lim_{t \to \infty} \frac{Q_l(t)}{t} = 0$.

To stabilize the queues, the long-term average of aggregated rates must satisfy:

$$\lim_{t \to \infty} \sum_{t'}^{t} y_l(t)/t \leq C_l, \forall l \tag{3}$$

Here the basic assumption is that the long-term average traffic load generated in the network can be handled by the capacity of the network, so that the network can be stabilized with proper rate control scheme. Otherwise, there is not much a transport layer rate control mechanism can do to avoid packet loss. This constraint is later relaxed into the objective of the minimization problem during the transformation, so that flows that use rates that exceeds link capacity is penalized.

3.2 Problem Formulation

We aim to devise an optimal source rate control mechanism to minimize overall per-packet delay in DCN with throughput guarantee. A stochastic delay minimization problem is formulated to encapsulate the above deadline (2) and network stability constraints (3).

$$\begin{aligned} \min_{\mathbf{x}(t), \mathbf{y}(t)} \quad & P_0(\mathbf{x}(t), \mathbf{y}(t)) \\ \text{subject to} \quad & y_l(t) = \sum_{s \in S(l)} x_s(t), \ \forall l \\ & \lim_{t \to \infty} \frac{\sum_0^t (\gamma_s(t) - x_s(t))}{t} \leq 0, \forall s \\ & \lim_{t \to \infty} \sum_{t'}^{t} y_l(t)/t \leq C_l, \forall l \\ & x_s(t) > 0, \ \forall s \end{aligned} \tag{4}$$

As mentioned above, the link capacity constraints are relaxed into the objective function to allow for temporary overloading of links. In the rest of this section, we apply the Lyapunov optimization framework to transform this problem to a convex problem, and then derive an optimal congestion window update function (see Section 3.5) based on the optimal solution to the transformed convex problem.

3.3 Transformation Using Lyapunov Optimization

Using the Lyapunov optimization theorem and the *drift-plus-penalty* method [9], the minimization of long term average are substituted with an equivalent convex minimization problem (5).

$$\begin{aligned} \min_{\mathbf{x}(t), \mathbf{y}(t)} \quad & \sum_s \{ V \sum_{l \in L(s)} d_l(y_l(t)) + Z_s(t) \gamma_s(t) / x_s(t) \\ & + \sum_{l \in L(s)} Q_l(t) x_s(t) \} \\ \text{subject to} \quad & y_l(t) = \sum_{s \in S(l)} x_s(t), \ \forall l \end{aligned} \tag{5}$$

where V is a non-negative weight that is chosen as desired to affect a performance tradeoff, and is set to be 1 for the rest of

the paper. In this transformation, the inequality constraints in (4) need to be transformed into virtual queues, $Z_s(t)$. These queues take the expected rate $\gamma_s(t)$ as input and the actual rate $x_s(t)$ as output. We have:

$$Z_s(t+1) = [Z_s(t) + \gamma_s(t) - x_s(t)]^+, \forall s$$

Virtual queues $Z_s(t)$ stores the difference in the expected transmission rate and actual transmission rate, and the queue lengths are essentially historical deviation from expected rates of the flows.

With the transformed problem, we developed an adaptive source rate control algorithm by greedily minimizing the upperbound of the Lyapunov drift.

3.4 Optimal Solution

By considering the properties of the optimal solution and the KKT conditions [4] of the above problem, we obtain a primal algorithm to achieve optimality for (5).

$$\frac{d}{dt}x_s(t) = (f_s'(x_s(t)) - \sum_{l \in L(s)} \lambda_l(t)) \tag{6}$$

where $f_s(x_s) = -Z_s(t)\gamma_s(t)/x_s(t) - Q_s(t)x_s(t)$, $\lambda_l(t) = d_l'(y_l(t))$, and $y_l(t) = \sum_{s \in S(l)} x_s(t)$.

We establish the stability and optimality of the rate update formula (6) by proving the following theorem.

THEOREM 1. *let*

$$\Pi(\mathbf{x}) = \sum_s f_s(x_s(t)) - \sum_s \sum_{l \in L(s)} d_l(\sum_{s \in S(l)} x_s). \tag{7}$$

$\Pi(\mathbf{x})$ *is strictly concave in* $\mathbf{x}$. *The unique solution* $\mathbf{x}^*$ *that maximizes* $\Pi(\mathbf{x})$ *is a stable point of the dynamic system, to which all trajectories converge.*

Proof: $f_s''(x_s) = -p_s''(x_s) = -2\frac{Z_s(t)M_s(t)/\tau_s(t)}{x_s^3} < 0$, for $x_s > 0$. Since $d_l(y_l)$ is convex and positive, thus $d_l''(y_l) > 0$. $\Pi''(\mathbf{x}) = \sum_s (f_s''(x_s) - \sum_{l \in L(s)} d_l(y_l)) < 0$, therefore $\Pi(\mathbf{x})$ is a concave function. Also, $d\Pi(\mathbf{x})/dx_s = f_s'(x_s) - \sum_{l \in L(s)} d_l'(y_l)$. $f_s'(x_s) = Z_s(t)M_s(t)/\tau_s(t)x_s^2 - Q_s(t)$ is an decreasing function, and approaches infinity as x goes to 0. On the other hand, $d_l(y_l)$ is strictly increasing, $d_l'(y_l) > 0$. It follows that there exist an unique value x_s such that $d\Pi(\mathbf{x})/dx_s = 0$ for $x_s > 0$.

$\Pi(\mathbf{x})$ is therefore strictly concave with an interior maximum. The maximal $\mathbf{x}^*$ is unique, and can be identified by $d\Pi(\mathbf{x})/dx_s = 0$.

$$\begin{aligned}\frac{d}{dt}\Pi(\mathbf{x}) = \sum_s \frac{\partial \Pi}{\partial x_s}\frac{d}{dt}x_s(t) = [\sum_s f_s(x_s(t)) \\ - \sum_s \sum_{l \in L(s)} d_l(\sum_{s \in S(l)} x_s)]^2 \geq 0\end{aligned} \tag{8}$$

which demonstrates that $\Pi(\mathbf{x}(t))$ is strictly increasing with respect to t, unless $\mathbf{x} = \mathbf{x}^*$. $\Pi(\mathbf{x})$ is therefore a Lyapunov function of the dynamic system, and the theorem follows. □

3.5 Optimal Congestion Window Update Function

As proven above, Formula (6) stabilizes and minimizes the per-packet delay of the network. With the optimal dynamics of the system determined, each flow should adjust their transmitting rate according to (6), which can be expressed as:

$$\frac{d}{dt}x_s(t) = (\Theta(\gamma_s(t), x_s(t)) - \sum_{l \in L(s)} (Q_l(t) + \lambda_l(t))) \tag{9}$$

where $\Theta(\gamma_s(t), x_s(t)) = \frac{Z_s(t)M_s(t)}{\tau_s(t)x_s^2(t)} = \frac{Z_s(t)\gamma_s(t)}{x_s^2(t)}$.

Let $\tau_s(t)$ be the RTT of flow s at time t. We can then derive the equivalent optimal window updating function:

$$\begin{aligned}W_s(t + \tau_s(t)) \leftarrow W_s(t) + \tau_s(t)(\Theta(\gamma_s(t), \frac{W_s(t)}{\tau_s(t)}) \\ - \sum_{l \in L(s)} (Q_l(t) + \lambda_l(t)))\end{aligned} \tag{10}$$

One way to interpret this result is to consider the 2 terms that constitute the difference between the two window sizes. The first (source term) is $\Theta(\gamma_s(t), x_s(t))$, which is an increasing function of γ_s, and a decreasing function for x_s. A large γ for a flow means that this flow have large remaining data and/or a urgent deadline. This term ensures that the flow will be more aggressive as its urgency (characterized by γ) grows. The second (network term), $\sum_{l \in L(s)}(Q_l(t) + \lambda_l(t))$, summarizes the congestion in the links along the path. If any of the links are congested, the source rate at each source that uses the link will curtail their transmission rate. With these two terms, our analysis above shows that this updating function will lead to a stable and minimal delay system.

4. PRACTICAL MCP ALGORITHM

The source term can be easily obtained by each source, but the network term is not straightforward. Since the sum of all prices, λ_l, and queue lengths, Q_l, are needed along the path, aggregated per-hop information is necessary for this optimal dynamics to work. For OpenFlow [8], the sum can be stored in an additional field in the packet header, and the switch adds and stores its own price and queue length to this field for every packet. However, current commodity switches are not capable of such operations. To devise a practical implementation, we use the readily accessible functionality in commodity switches, the ECN mechanism.

4.1 ECN-based Window Update

At the beginning of the transmission, the initial window size is set to $RTT \times \gamma_s(0)$. $\gamma_s(0)$ is the average transmission rate that flow s attempts to achieve throughout its life time, and therefore is a good starting point. In fact, as proven above, MCP is guaranteed to achieve the optimal congestion window size for any starting point. The RTT is an estimated value from the handshake packets.

The focus of our approximation is the metric at bottleneck link of each flow, as the source term is self-maintained in each

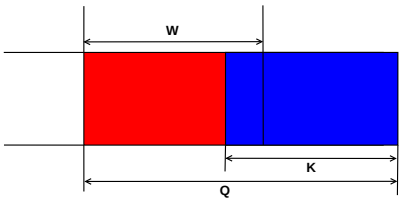

Figure 2: Queue length approximation

source. We approximate the second term in (10), which is the sum of two single terms: the queue length of this bottleneck link, and also its link price.

We denote F as the fraction of packets that were marked in the last window of packets, and we do not compute the moving average as in DCTCP [1] since we are interested in attaining an estimation that is closest to the instantaneous queue length. F is updated for every window of packets.

Figure 2 demonstrates the method to estimate the queue length from F. The fraction marked by ECN is less than the red portion, therefore $F \leq \frac{Q-K}{W}$, and $Q \geq K + F \times W$. We take the lower bound as the estimate, $\hat{Q}$, thus $\hat{Q} = K + F \times W$.

For the price, we adopt the M/M/1 queue delay formula [7], $d(y) = 1/(C-y)$, where y is the arrival rate to the link, and C the link capacity. Therefore the price of the link is proportional to the derivative of the delay function, $d'(y) = (C-y)^{-2}$, for different weights assigned by sources. The arrival rate can be directly obtained by two consecutive queue estimations at the source. The estimated $\hat{y}(t) = \frac{\hat{Q}(t) - \hat{Q}(t-\tau_s(t))}{\tau_s(t)}$.

The window update function therefore becomes

$$W_s(t+\tau_s(t)) \leftarrow W_s(t) + \tau_s(t)(\Theta(\gamma_s(t), \frac{W_s(t)}{\tau_s(t)}) - (K + F_s(t)W_s(t) + \lambda(t))) \quad (11)$$

where $\lambda(t) = (C - \frac{F_s(t)W_s(t) - F_s(t-\tau_s(t))W_s(t-\tau_s(t))}{\tau_s(t)})^{-2}$

Every source needs only two consecutive window sizes and fractions of ECN marked packets to compute its window update. We demonstrate that the gap between this approximation and the optimal scheme is close in the evaluation. In general, MCP changes the window update behavior of traditional TCP while preserving its functionalities for reliable transmission.

4.2 Handling Failures

We assume the long-term average load of the network does not exceed the physical network capacity, i.e., constraint (4). However, in practice, link or other failures may occur, reducing the capacity and rendering this assumption invalid. The virtual queue $Z_s(t)$ stores the difference between the actual rate and the expected rate, i.e., the rate that is 'left over' in the transmissions. In case of failure, $Z_s(t)$ may grow rapidly, and forces the source to send at a higher rate, overloading the network. To prevent this, we stipulate that the source will abort the transmission of a flow when $Z_s(t) > \max_{l \in L(s)} C_l$, as it implies that even the largest link capacity along its path is no longer sufficient for the flow to finish before deadline. Aborting flows that are impossible to meet their deadlines gives more opportunities for other flows to meet their deadlines.

5. PRELIMINARY NUMERICAL RESULTS

As preliminary study, we compare MCP against DCTCP and D2TCP using numerical simulations in Matlab, and our immediately next step is packet-level simulations and real implementation.

The simulation setting is as follows. We use a 128-node three level fattree to simulate a simple DCN environment. The link capacity is 10G for ToR (Top-of-Rack) to aggregation link, and 40G for aggregation to core link. We generate three types of traffic, i.e., query (2KB to 20KB), short (100KB to 1MB), and long flows (1MB to 100MB). The flow sizes are drawn from an exponential distribution with mean equals to 1MB, and capped at 100MB. The deadlines are calculated based on the flow types and the corresponding share of the bandwidth. Each node will abort its flows as soon as the flows fail to meet deadlines. The load of the network is adjusted by setting the arrival rate of the flows at the nodes. To compare with DCTCP and D^{2}TCP, we set their ECN marking threshold as 64 packets. The simulation runs for 30 seconds. The optimal scheme is denoted by "MCP-Opt", and its congestion window update function follows (10). The practical approximation (11) is denoted as"MCP-Approx".

We first examine the deadline miss rate. It is obvious that MCP outperforms DCTCP and D2TCP for all flows. For query and short flows, all schemes performs similarly under 10% maximum load. However, the rates diverge as load increases. For query flows, MCP-Opt and MCP-Approx manage to maintain deadline miss rate close to zero, with lower than 2% even at maximum load. For short flows, the deadline miss rates of the two MCP schemes are consistently low and close to zero, as compared to around 40% for DCTCP and 20% for D2TCP. For long flows, MCP again demonstrates advantages over D2TCP and DCTCP. Even with the loads increase to maximum, less than 4% of long flows miss their deadlines for MCP. On the other hand, D2TCP and DCTCP cannot provide enough bandwidth for the long flows with various deadlines properly, resulting in 50% and 70% deadline missed at 80% maximum load. As shown across Figure 3 (a, b, c), MCP is able to meet almost all deadlines, and to achieve network stability under heavy loads, as a result of MCP's ability to find and sustain the *right* rate for every flow. However, DCTCP and D2TCP are not able to precisely capture the rate requirements of flows.

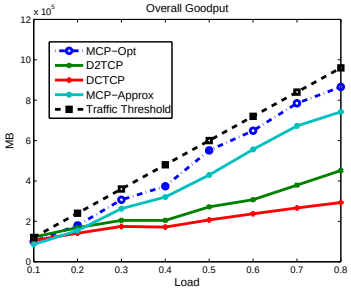

Figure 4: Numerical simulation results: overall goodput

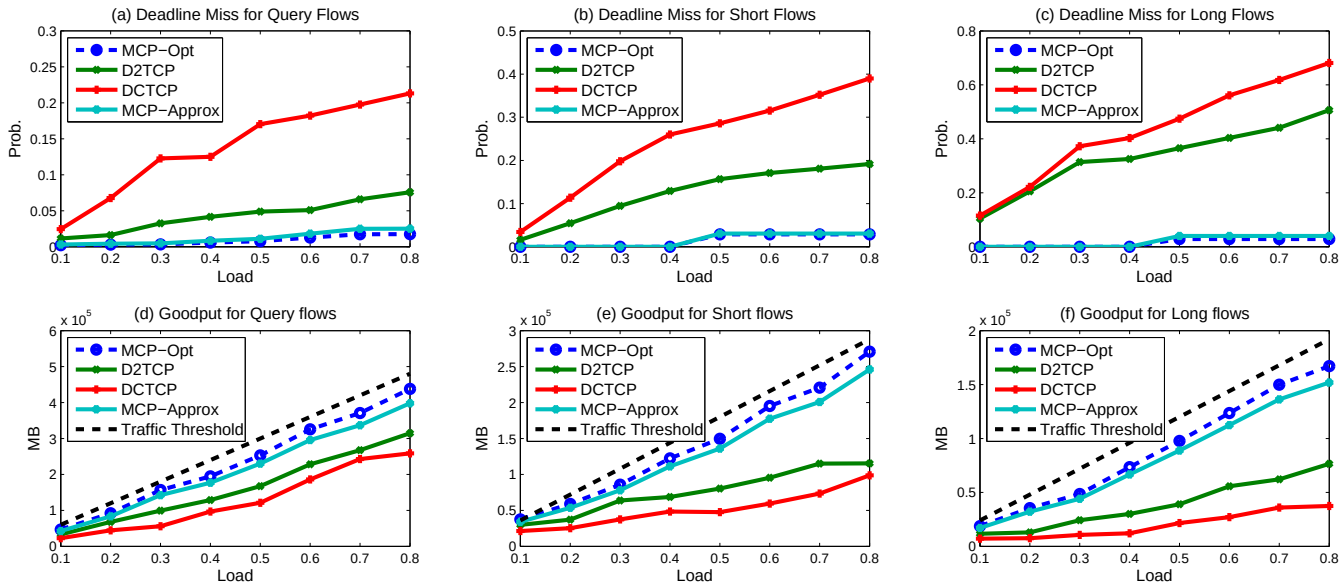

Figure 3: Numerical simulation results: deadline miss rate and goodput

In Figure 3 (d, e, f), we compare the goodput of the protocols for 3 traffic types. Goodput is defined as the amount of data transmitted successfully within the deadlines. The black dotted line is the amount of traffic sent from the nodes for each load level (denoted as "Traffic Threshold"), and the performance of MCP-Opt and MCP-Approx is close to this traffic threshold for all traffic types. On the other hand, although D2TCP and DCTCP can achieve most of the traffic at 10% maximum load, these two protocols fall off at high load levels for all traffic types. As for the overall performance across all traffic types (Figure 4), MCP satisfies almost all traffic requirements for having close-to-zero deadline miss rates.

6. RELATED WORKS

DCTCP ensures low latency for short flows and high throughput for long flows, yet it shows no concern for meeting deadlines. The bandwidth is shared equally among flows with vastly different deadlines, resulting in higher deadline miss rate. In D2TCP, when congestion occurs, far-deadline flows back off aggressively, while near-deadline flows back off only a little or not at all. This approach is problematic in handling flows with rate requirements, for example, long flows with deadlines. MCP, on the other hand, determines the appropriate rate based on the demand rate of the flow and the network congestion, and can accommodates more generic traffic types.

D3 [11] tackles the missed deadlines in DCN using a centralized and proactive approach, which requires the switches to be modified significantly [10]. As a comparison, MCP is a distributed approach, which is more suitable in a highly dynamic environment such as DCN.

In the attempt to achieve ultra-low latency, HULL [2] sacrifices the transmission rate and flow completion time for large flows by reserving bandwidth headroom. Since DCTCP is a key component of HULL, HULL shares the same problems.

Recently, pFabric [3], by decoupling flow scheduling and rate control in DCN transport layer, has shown near-optimal performance for high priority flows and high utilization in simulations. As flow scheduling is performed in the network fabric, pFabric requires clean-slate design in the transport layer, and non-trivial hardware modification. As a comparison, MCP requires no hardware modifications, and is readily deployable.

Similar to pFabric, PDQ [6] minimizes FCT by means of preemptive flow scheduling. Besides difficulties in implementation, PDQ potentially suffers from large overhead, as switches need to reach consent to set the rates of the flows. We, however, believe that an implicit and reactive scheme is more suitable for DCN, since the environment is highly dynamic.

7. CONCLUSION AND FUTURE WORK

We have proposed MCP, a novel transport protocol to minimize packet delay while providing guaranteed rates to meet flow deadlines. MCP follows the line of TCP variants like DCTCP and D2TCP that requires no switch hardware modification and support incremental deployment. To design MCP, we formulate and solve a stochastic delay minimization problem, and adapt the derived optimal algorithm to a practical MCP protocol, which fully utilizes currently available switch functionality, i.e., ECN, to determine the *right* transmission rate for every flow. Theoretical analysis shows the validity of MCP. Preliminary results show the potential of MCP in terms of deadline meet rate and goodput.

We anticipate two directions for future research. Theoretically, we seek to improve the system model to incorporate more practical setting, as well as analyze performance bounds of MCP. Practically, we will further polish MCP design and its practical approximation, evaluate it using packet-level simulators, and then implement and experiment with MCP in a real DCN testbed environment.

8. REFERENCES

[1] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *Proceedings of the ACM SIGCOMM 2010 conference*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 63–74. [Online]: http://doi.acm.org/10.1145/1851182.1851192.

[2] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: trading a little bandwidth for ultra-low latency in the data center," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 19–19. [Online]: http://dl.acm.org/citation.cfm?id=2228298.2228324.

[3] M. Alizadeh, S. Yang, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "Deconstructing datacenter packet transport," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XI. New York, NY, USA: ACM, 2012, pp. 133–138. [Online]: http://doi.acm.org/10.1145/2390231.2390254.

[4] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[5] S. Floyd, "TCP and explicit congestion notification," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 5, pp. 8–23, 1994.

[6] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 127–138, Aug. 2012. [Online]: http://doi.acm.org/10.1145/2377677.2377710.

[7] L. Kleinrock, *Theory, volume 1, Queueing systems*. Wiley-interscience, 1975.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[9] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 1, pp. 89–103, 2005.

[10] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter tcp (D2TCP)," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 115–126, Aug. 2012. [Online]: http://doi.acm.org/10.1145/2377677.2377709.

[11] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: meeting deadlines in datacenter networks," in *Proceedings of the ACM SIGCOMM 2011 conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 50–61. [Online]: http://doi.acm.org/10.1145/2018436.2018443.