Implementation and end-to-end Evaluation of the HULL Architecture

Prasopoulos Konstantinos
School of Computer & Communication Sciences, EPFL, Switzerland

***Abstract*—The abstract**

I. Introduction

(temporary)

- Data ceter traffic characteristics - the need for low queuing latency
- Briefly - the approach HULL takes
- The "hole" that this project fills [1]

II. System Design and Implementation

- Design overview
- —DCTCP [2]
- —PQs
- —Pacer
- NS2 Implementation
- —DCTCP
- —PQs
- —Pacer

III. Results

This section presents the results obtained by simulating the HULL architecture under a variety of workloads using ns-2. First, this implementation is compared to the experimental findings of the original paper. Second, HULL's performance is evaluated under two types of partition/aggregate workloads. The architecture is compared to TCP Reno and DCTCP (with the recommended parameters and in a more aggressive configuration). The network topology simulates a single rack with a configurable number of servers connected to a switch. In all simulations, one of the servers behaves as a client and issues requests to the other servers. The responses cause congestion at the client's switch port. Table I summarizes the system parameters which hold unless specified otherwise. The MTU is set to $950 Bytes$ so that the phantom queues configured with a $1KB$ threshold (as in the original paper) do not mark all maximum sized packets even if there is no other traffic (the PQ increments the virtual queue length upon receiving a packet as described in 5.1 of [3]).

Network	Link Speed = $1Gbps$, Link Latency = $5\mu s$, Port Queue = $500pkts$
TCP	Max. Window = 1256, Max. ssthresh. = inf, MTU = $950bytes$, Min. RTO = $10ms$, Nagle: Disabled
DCTCP	g = $1/16$, K = $30KB$
Pacer	Bucket Depth = $3KB$, η = 0.125, β = 16, p_a = 0.125, T_i = $10ms$
Phantom Queue	Drain Rate = 95%, Threshold = $1KB$

Table I: General System Parameters

A. Implementation Evaluation

In order to make sure this implementation of HULL behaves as intended, two of the experiments of the original paper where approximately reproduced. The results were not expected to be identical as there is a wide range of differences between the original experimental setup and this simulation. Some of those differences are: 1) ns-2 uses fixed size switch buffers compared to the shared, dynamically allocated memory of the Broadcom Triumph2 switch, 2) software overhead or features like Interrupt Coalescing are not simulated at the servers, 3) the exact details of the system differ (e.g. propagation delay, TCP configuration etc.) are not known. Nevertheless, the fundamental properties are the same and HULL generally manages to keep the congested port's buffer mostly empty by sacrificing some of the throughput of the large flows.

***Static Flow Experiments*:** This experiment aims to measure throughput and the congested port's queue occupancy under heavy and continuous load. To this end, a client sends requests for the same file to a static number of servers which in turn respond by sending data as fast as possible. The congested port's queue length is sampled every 1ms and is used to calculate the resulting queuing latency. As in the original paper, four schemes are compared: (1) TCP with droptail buffers, (2) DCTCP with a 30KB marking threshold, (3) DCTCP with a 6KB marking threshold along with pacing and (4) DCTCP with both pacers and phantom queues with the marking threshold at 1KB and the drain rate at 900Mbps (the original paper uses a 950Mbps drain rate but it was quite ineffective in this experiment). In scheme 4, it is the PQs and not the switch that are responsible for setting the packets' CE field.

The simulation results can be seen in figure 1b. In general, HULL reduces the switch latency (i.e. the queue occupancy) by two orders of magnitude compared to TCP and by one compared to DCTCP as is the case in the original results

		Switch Latency (μs)		900B FCT (μs)		10MB FCT (ms)	
		Avg	**99th**	**Avg**	**99th**	**Avg**	**99th**
20% Load	TCP-Droptail	83.0	2656.3	134.8	2507.5	**99.4**	256.5
	DCTCP-30K	6.4	132.8	64.8	892.2	99.7	248.6
	DCTCP-6K-Pacer	3.8	121.1	60.8	794.2	101.7	**242.8**
	DCTCP-PQ950-Pacer	**0.3**	**8.3**	**41.3**	**122.7**	205.4	290.3
40% Load	TCP-Droptail	370.7	3583.3	436.7	3628.1	**128.4**	379.1
	DCTCP-30K	27.2	215.8	81.9	867.2	131.5	**371.8**
	DCTCP-6K-Pacer	15.6	166.3	77.2	862.7	138.3	376.0
	DCTCP-PQ950-Pacer	**2.5**	**42.2**	**44.2**	**104.7**	236.6	404.9
60% Load	TCP-Droptail	823.5	3741.4	910.4	3828.9	**197.6**	722.2
	DCTCP-30K	57.4	239.0	123.4	923.1	210.5	720.3
	DCTCP-6K-Pacer	29.5	182.5	106.2	917.05	219.8	**718.0**
	DCTCP-PQ950-Pacer	**12.0**	**106.1**	**55.6**	**150.9**	310.3	731.5

Table II: Dynamic flow simulation results. The displayed load level refers to the traffic level on the client-switch link.

(figure 1a). The present results differ from the original in the following ways: 1) The complete scheme (DCTCP-PQ950-Pacer) converges to the DCTCP-6K-Pacer at lower flow counts, (2) The latency of the complete scheme is higher with 2 flows than with 3, (3) The complete scheme has the same throughput as the rest for flows 4 to 8, (4) TCP is irregular with 2 flows. The last one is not important and is caused by the fact that one of the flows manages to starve the other early on - a behavior not encountered again.

The first three differences are all caused by the way the $5\mu s$ per link propagation delay used in the simulation affects the round trip time (RTT). Given that the sending rate of TCP-like protocols is approximately $congestion\ window/RTT$ and the RTT is about $40\mu s$ ($20\mu s$ to transmit a 1KB packet twice and $20\mu s$ for the ack to return due to propagation delays - not accounting for queuing delay), each additional window length unit adds $MTU/RTT \approx 200Mbps$ to the throughput of the flow. The minimum window size used during the simulation was 2 and at this size each flow contributes a minimum of $400Mbps$. With 3 flows, the total of $1200Mbps$ consistently exceeds the drain rate of the phantom queue between the switch and the client and the sending window is fixed at 2 packets. However, the sender practically ignores the marked acks as the sending window is already fixed at its minimum size. With 2 flows present, the average window length is 3.4 packets and the senders are probing for additional bandwidth and reacting to congestion normally (hence the higher latency variation).

Figure 1c further illustrates this point. The experiment is repeated with $100\mu s$ link propagation delay instead of $5\mu s$. It is impportant to note that this is a far-fetched scenario in which very large flows are synchronized and this issue does not affect the rest of the results (at $5\mu s$).

Dynamic Flow Experiments: In this experiment, small $900Byte$ requests are combined with large $10MB$ ones as follows: The client sends requests for small and large files at intervals drawn from two different Poisson distributions configured based on the average target load on the congested switch-client link. For each request, the client asks the servers in a round-robin fashion. The ratio between the number of small and large requests is 4 to 1 while the throughput ratio is 1 to $2,222$. The queuing latency at the switch's port facing the client is calculated based on the momentary queue occupancy. Finally, the flow completion times (FCT) are measured for both the small and large flows.

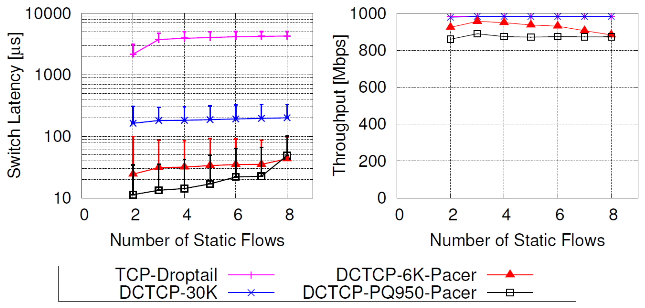
The simulation results for three levels of link utilization are shown in table II. The results follow a trend similar to the original paper. Considering the 99^{th} percentile of measurements, the simulation of HULL shows an average switch latency reduction of 98.4% compared to TCP and of 73.4% compared to DCTCP (98.3% and 88.6% in the original paper respectively). Concerning the flow completion times of the small flows, the simulation results indicate an average FCT reduction of 96.1% compared to TCP and of 85.8% compared to DCTCP (89.1% and 60.2% in the original paper). Finally, the FCT of the large flows increased by 5% compared to TCP and by 6.4% compared to DCTCP (20.3% and 35.8% in the original paper). However, the mean FCT for the large flows increases by 69.8% based on the simulation but only by 34.8% based on the original paper's results. Given that the simulation results appear to trade-off bandwidth for latency compared to the original paper, it is reasonable to assume that HULL could be configured less aggressively to account for the differences between the simulation and the physical environment.

B. Partition/Aggregate Workloads

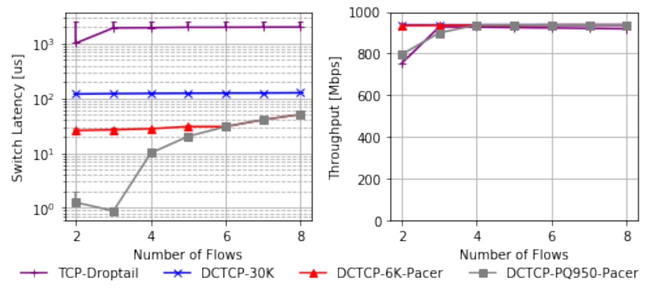
Search: In this section, HULL is evaluated against a web search-like, partition/aggregate workload. The client sends $3500Bytes$ queries at Poisson intervals to 10 servers. Each server processes the query for a duration drawn from:

$$Service\ time = 180 * gamma(0.7, 20000) + 10000\ [ns]$$

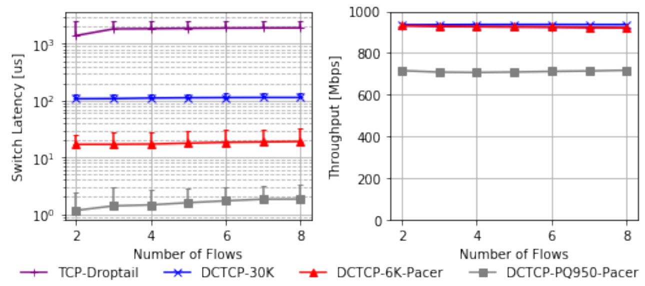
where $gamma(0.7, 20000)$ is the $gamma$ distribution with parameters $\kappa = 0.7$ and $\theta = 20000$. The overall mean service time is $2.53\ miliseconds$. Each server's response is $2800Bytes$. A query is considered complete when the slowest server responds. Each server processes the queries sequentially (not a significant factor at this query rate - 30%



(a) Original Results [3]



(b) Results with $5\mu s$ link propagation time



(c) Results with $100\mu s$ link propagation time

Figure 1: Left: Queuing latency at the congested client switch port (log scale). Right: Aggregate throughput of all flows. The error bars represent the 99^{th} percentile.

server occupancy). The workload parameters are based on the OLDISIM simulator [4].

The search workload was also combined with the background dynamic workload described in section III-A. The simulation was run at four different levels of average background traffic on the congested $1Gbps$ client-switch link: (1) no traffic, (2) $200Mbps$, (3) $400Mbps$, (4) $600Mbps$. The search workload generated approximately $12,000$ queries and an average of $26.5Mbps$ on the same link.

The results are displayed in figure 2. Starting from the rightmost chart, when the background traffic is a reasonable $200Mbps$, HULL manages to reduce the 99^{th} percentile of queuing latency to $10\mu s$ - one order of magnitude lower than DCTCP and two lower than TCP. However, this sub-$1ms$ reduction does not translate to a measurable improvement to the 99^{th} percentile of the query completion times (leftmost chart) which hover around $30ms$ and are dominated by the tail of the servers' processing time. The same applies to the 99^{th} percentile of the average server response time which can be seen in the second chart. When it comes to higher levels of background traffic, both DCTCP and HULL outperfrom TCP on both the 99^{th} percentiles and the means (not shown) by an appreciable margin. The third chart shows the average number of servers which respond within a $25ms$ (in the worst 1% of queries -worst defined by the number of servers that respond) and thus reflects the quality of the search if such a deadline was used. Both HULL and DCTCP have tight tail latencies and the quality is minimally affected by the background traffic.

It is worth mentioning that the pacer successfully selects which flows need pacing. For example, with $200Mbbps$ of background traffic, the pacer attached to one of the servers decided to associate the search flow 1695 times out of the 48000 search packets (3.5%) and the background flow $12,071$ times out of the $260,000$ packets (4.6%). This count includes the times a flow was already being paced - in that case the disassociation timer was reset. The difference between the flows is that the search flow was disassociated 1396 times (82%) while the background flow was only disassociated 94 times (0.8%). Essentially, even if a search query was paced, the next one was unlikely to be so while every background $10MB$ transfer was paced.

Low latency Search: The conclusion that can be drawn from the previous workload is that while HULL outperforms TCP, the additional $100\mu s$ of queuing time saved compared to DCTCP need an appropriate, very low latency application, in order to have an impact. Such a partition/aggregate workload is simulated in this section. It is similar to the previous search workload but the size of the queries and the responses is now $100Bytes$ and critically, the per-server service time is $100\mu s$. Approximately 50000 queries are generated along with an average of $4Mbps$ on the congested client-switch link. These parameters approximate an in-memory based search workload (e.g. [5]). The results are presented in figure 3. Across the scenarios that involve background traffic, HULL reduces the 99^{th} percentile of the query completion time by 84% compared to TCP and by 32% compared to DCTCP. Additionally, HULL offers perfect response quality with a strict $1ms$ response deadline.

IV. Conclusion

References

[1] J. Dean and L. A. Barroso, “The tail at scale,” *Commun. ACM*, vol. 56, no. 2, pp. 74–80, Feb. 2013. [Online]. Available: http://doi.acm.org/10.1145/2408776.2408794

[2] 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]. Available: http://doi.acm.org/10.1145/1851182.1851192

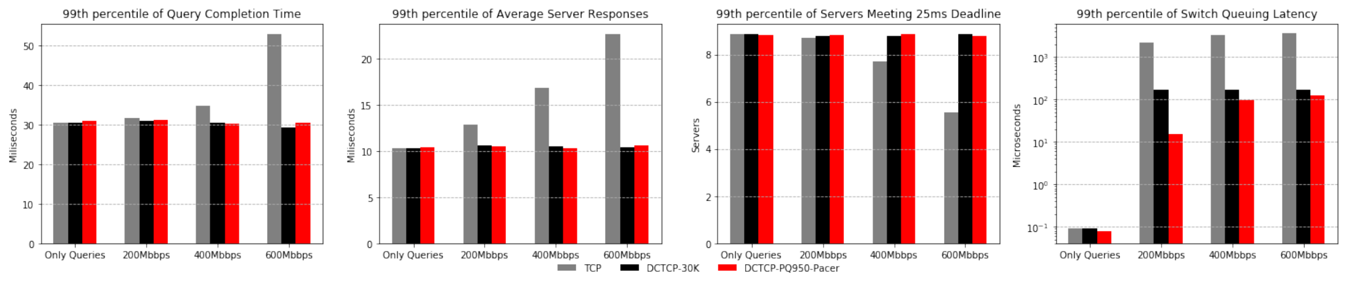
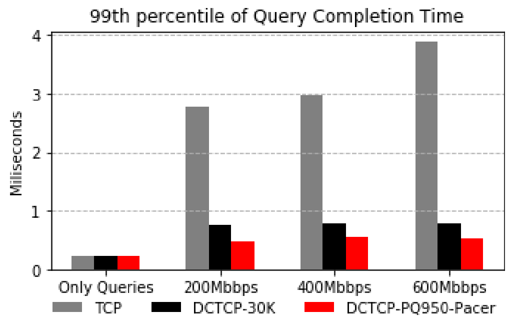
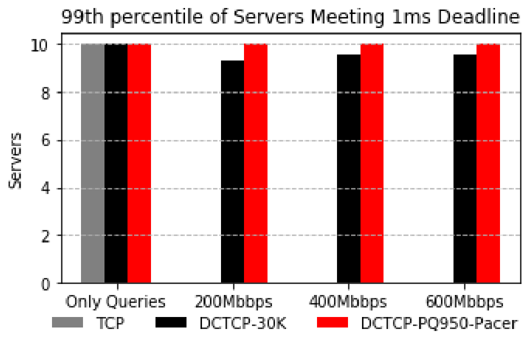


Figure 2: The x-axis varies based on the amount of background traffic along with the partition/aggregate workload (from none to 60% of the link capacity). (a): 99th percentile of query completion times (defined by the slowest server response), (b): 99th percentile of the average server response time (ignores application level variability), (c): 99th percentile of the number of servers that respond within a $20ms$ deadline (worst cases), (d): 99th percentile of the queuing induced latency at the congested client-switch port buffer.



(a) 99th percentile of query completion times (defined by the slowest server response).



(b) 99th percentile of the number of servers that respond within a $1ms$ deadline (worst cases).

Figure 3: The x-axis varies based on the amount of background traffic along with the partition/aggregate workload (from none to 60% of the link capacity).

[3] 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]. Available: http://dl.acm.org/citation.cfm?id=2228298.2228324

[4] “Oldisim,” https://cloudplatform.googleblog.com/2015/03/benchmarking-web-search-latencies.html.

[5] Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, “Characterizing facebook’s memcached workload,” *IEEE Internet Computing*, vol. 18, no. 2, pp. 41–49, Mar 2014.

APPENDIX