Technical Report

UCAM-CL-TR-933
ISSN 1476-2986

Number 933

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

An Evaluation of NDP performance

Noa Zilberman

January 2019

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

An Evaluation of NDP Performance

Noa Zilberman

Abstract

NDP is a novel data centre transport architecture that claims to achieve near-optimal completion times for short transfers, and high flow throughput in a wide range of scenarios. This work presents a performance evaluation of NDP, both on the simulation and the hardware level. We show that NDP's implementation achieves lower switch throughput than simple TCP, and that the simulated performance is highly dependent on the selected parameters.

1 Introduction

In 2017, Handley *et al.* proposed a new data centre protocol, NDP [5]. NDP aims to achieve both low latency and high throughput by combining several concepts. First, it allows flows to start sending immediately at full rate, without connection setup. Second, it load-balances using per-packet multipath. Third, it trims packets in the switch, when the switch's queue fills up. The trimming leads to loss of payload but no loss of metadata. The NDP protocol is then built on top of these concepts.

The NDP design claimed the following contributions:

- Better short-flow performance than DCTCP or DCQCN.
- Greater than 95% of the maximum network capacity in a heavily loaded network with switch queues of only eight packets.
- Near-perfect delay and fairness in incast scenarios.
- Minimal interference between flows to different hosts.
- Effective prioritization of straggler traffic during incast

The authors of NDP have released the evaluation environment of NDP [4], including both the source code of the switch design and the simulation environment. As part of the Stardust project [15], we have used the NDP evaluation environment to study different aspects of data centre network (DCN) performance, and came across some surprising results. In this paper, we summarize these results. In particular, we find that:

- The NDP switch fails to achieve full line rate, for certain packet sizes.
- The throughput of the NDP switch is inferior to TCP.

- The throughput achieved by NDP in simulation is highly dependent on assumed parameters.

- In NDP, flows may timeout and never complete, leading to a significant performance loss.

The work presented in this paper does not represent a comprehensive study of NDP, however it provides greater depth into understanding its performance and limitations.

2 Evaluation Environment

The evaluation environment of NDP is open source and available in [4]. The environment contains four components: a simulation environment, an implementation of NDP switch in P4, an implementation of NDP switch on the NetFPGA platform, and an implementation of the host side.

In this paper we use for the evaluation NDP repository commit 69d27f53c90f441ae1562b47 15d084cebe544b3a from January 8th, 2018. The NDP environment is relatively easy to use, and is provided with wrappers for most of the tests, enabling reproducibility of the tests conducted by the authors of NDP. The code itself is not well documented.

The implementation of the P4 switch targets only the P4 behavioural model, and in [5] the authors omitted evaluation results. It was not evaluated as part of this work, neither for performance nor for correctness.

The host side implementation was released after the completion of this evaluation work.

The code used for the evaluation of NDP is available as a fork of the original NDP repo, under `https://github.com/noaz/NDP-eval`. For some of the tests we had access to code not openly released by NDP authors but made available to us. This code is not included in our public repository.

2.1 Hardware Environment

The Implementation of NDP on NetFPGA SUME [14] is based on the NetFPGA Reference Switch design. The NDP design supports both NDP and non-NDP traffic, with non-NDP traffic passing through the design as in the Reference Switch, while NDP traffic is treated according to the NDP protocol, i.e. adding support for trimming and priority queues. In addition, the Input Arbiter of the switch is modified, supporting deficit round robin (DRR), instead of the reference round robin (RR).

In this work, we synthesize the NDP switch using NetFPGA-SUME release 1.7.1 (71c98c8b d84af624c3d59e09d9152f0210645c62) from December 14th, 2017. We evaluate the performance of NDP compared with the NetFPGA Reference Switch, running traffic through the both designs.

We evaluate the switches using OSNT [1], an open source network tester, implemented over the NetFPGA SUME platform. We use OSNT SUME release 1.7.0 (e116fdba2a3cf791d44f748 7d3b9b973cf08835a) from July 14th, 2017, and use the bitfile released by the OSNT team (`https://www.cl.cam.ac.uk/research/srg/netos/projects/netfpga/bitfiles/`

`OSNT-SUME-live/osnt-extmem-20170810.bit`). We use OSNT only as a traffic generator, and not for latency measurements.

Our setup is composed of two identical NetFPGA SUME boards, one configured as OSNT, and the other as the device under test. The boards are hosted within two identical i7-6700K machines running at 4GHz and using Ubuntu 14.04, yet the host has no effect on the test.

2.2 Simulation Environment

The simulation environment is based on htsim [10]. The simulator and its codebase are provided as part of the NDP repository [4]. The code contains implementations of TCP NewReno (not SACK), a version of MPTCP, DCTCP, and PFC/DCQCN. The DCQCN implementation is based on the DCTCP code and is window based rather than rate based.

We run all the simulations on a Xeon E5-2660 v4 server, using 256GB of DDR4-2400, running at 3.2GHz. The operating system used is Ubuntu 14.04, kernel version 3.13.0-32-generic.

3 Hardware Performance

In this section we evaluate the performance of the NDP switch, in comparison with the NetFPGA Reference Switch. We focus our evaluation on the throughput of the design, and do not measure the latency through the device.

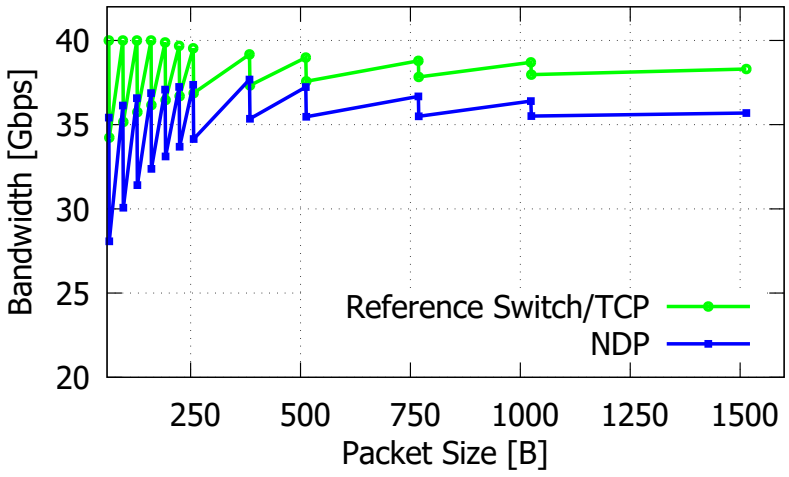
The NetFPGA SUME data path is 32B wide (256 bit), and every packet starts a new transaction on the data bus, meaning that different packets do not share the data bus on the same clock cycle. The data path width dictates that it will take at least 2 clock cycles to pass a 64B packet through a certain stage of the design, and at least 3 clock cycles for a 65B packet. In general the number of cycles per packet is:

$$Cycles \geq \left\lceil \frac{SIZE\_BYTES}{32} \right\rceil$$

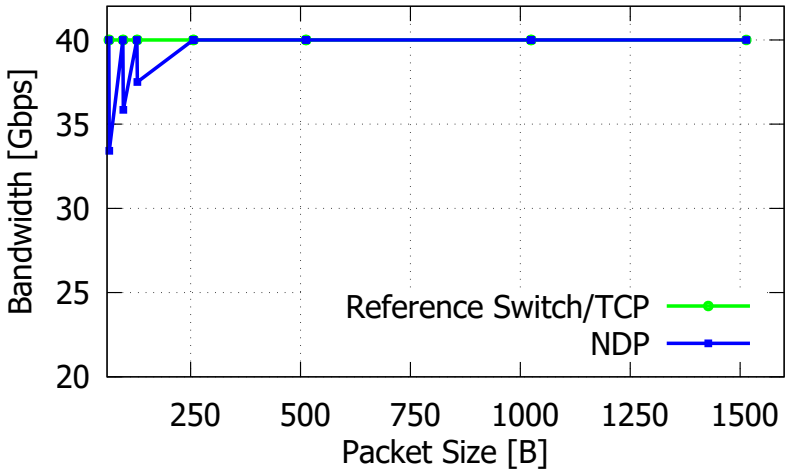
NetFPGA SUME Reference Switch supports $4 \times 10GE$, which is roughly 59.52Mpps for 64B packets. As 64B packets take 2 clock cycles, the minimum core frequency required is 120MHz. For 65B packets, it is expected that full line rate will be achieved at a core clock frequency of roughly 180MHz ($58.8Mpps \times 3cycles$), and so on.

In the following experiment, we study the throughput of the NetFPGA Reference Switch and the NDP switch for different packet sizes and different core clock frequencies. We connect $4 \times 10GE$ ports between OSNT and the NetFPGA board, and use OSNT to generate traffic at full line rate. In every experiment, only a single packet size is used. We send either TCP or NDP headers, and use the NDP packets generation script written by NDP's authors. The experiment is repeated multiple times, and the results reproduce with minor variations (not documented below). We note that in some of the experiments the setup crashes, regardless of the device under test, and we believe the source of the problem is OSNT.

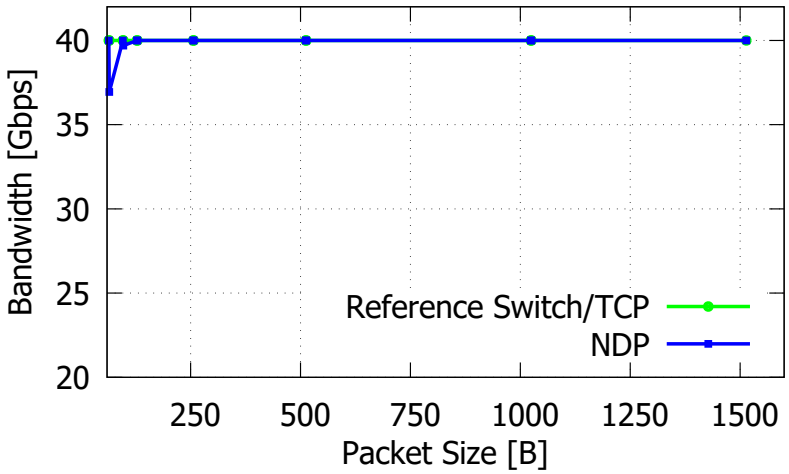
In each iteration, we send a hundred million packets from each of the four ports on OSNT, and use the NetFPGA SUME counters to check how many packets successfully passed through



(a) 150MHz



(b) 180MHz



(c) 200MHz

Figure 1: The throughput of the evaluated switches running at different core clock frequencies.

the data path, and were sent back to OSNT. We compare the counters in multiple places in the design, checking for packet drops and unrelated issues, e.g., 10G port limitations. Our baseline for the comparison is the Reference Switch: NDP is expected to perform better than the reference design, but problems that exist in the reference design, such as outside the data path, are not expected to be fixed by NDP and are not considered an issue with NDP.

We start our experiments at 120MHz, however we find that due to a limitation of the NetFPGA 10G ports this frequency is not practical [7]. We then benchmark the design at 150MHz, 180MHz and 200MHz. Both designs are expected to support full line rate using the last two frequencies. We scan a range of packet sizes: 64B, 65B, 96B, 97B, 128B, 129B, 160B, 161B, 192B, 193B, 224B, 225B, 256B, 257B, 384B, 385B, 512B, 513B, 768B, 769B, 1024B, 1025B, 1514B. The choice of packet sizes is on 32B granularity: a perfectly aligned packet size, followed by a misaligned packet size.

The results of our evaluation are presented in the following three figures: Figure 1(a) shows the performance using a core clock frequency of 150MHz, Figure 1(b) shows the results using 180MHz clock, and Figure1(c) shows the performance using 200MHz clock. As the results show, the Reference Switch consistently outperforms the NDP design, using NDP packets. Not only it achieves higher throughput at 150MHz, but it also achieves full line rate for all packet sizes at 180MHz, whereas NDP fails to achieve full packet rate for some packet sizes (65B, 97B) even at 200MHz.

As far as the authors of this paper can tell, the reason for NDP's performance loss is that the design is not fully pipelined, using a state machine that requires more clock cycles than actual packet length. NDP's authors confirmed this performance loss is expected in their design.

We have attempted to run a second test, in which we evaluate the performance of the switches based on packet size distributions from real traces [12]. Unfortunately, the NDP design continuously crashed and we were not able to complete the experiments. This crash is not caused by OSNT, and evaluation of the Reference Switch using the same traces was successful, as documented in [15].

4 Simulated Performance

The second part of our evaluation focuses on the simulation environment of NDP. We use the simulation environment "as is", except for the minimum amount of changes required to evaluate a specific aspect, e.g., setting the packet size or changing packet size distribution.

There is an important note to remember when considering all the reported results, which is that this evaluation explores both the simulation framework and the algorithms, thus some results that may seem surprisingly good, bad or unexpected, may be a result of the implementation of the simulation model, or the implementation of the simulator, rather than an indication of an algorithms performance. The evaluation, however, can either support our confidence in the results or discourage such confidence.

4.1 Per flow throughput

We reproduce the per-flow throughput experiment from [5], using the original scripts, and receive similar results. The experiment uses a 432-node fat-tree configuration, where each server

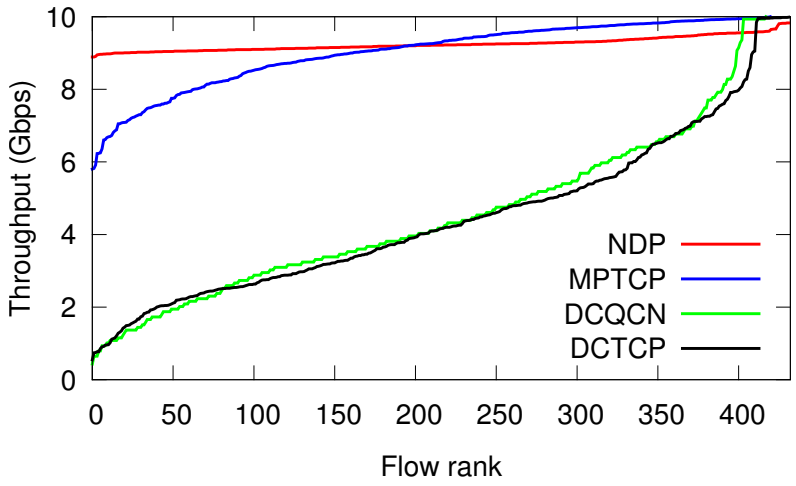


Figure 2: Per-flow throughput, permutation traffic matrix, 432-node Fat-Tree.

has a single long-running connection to another random server, and each server has exactly one incoming connection. The server exchanges single size packets, originally a constant 9000B. The reproduced results are presented in Figure 2.

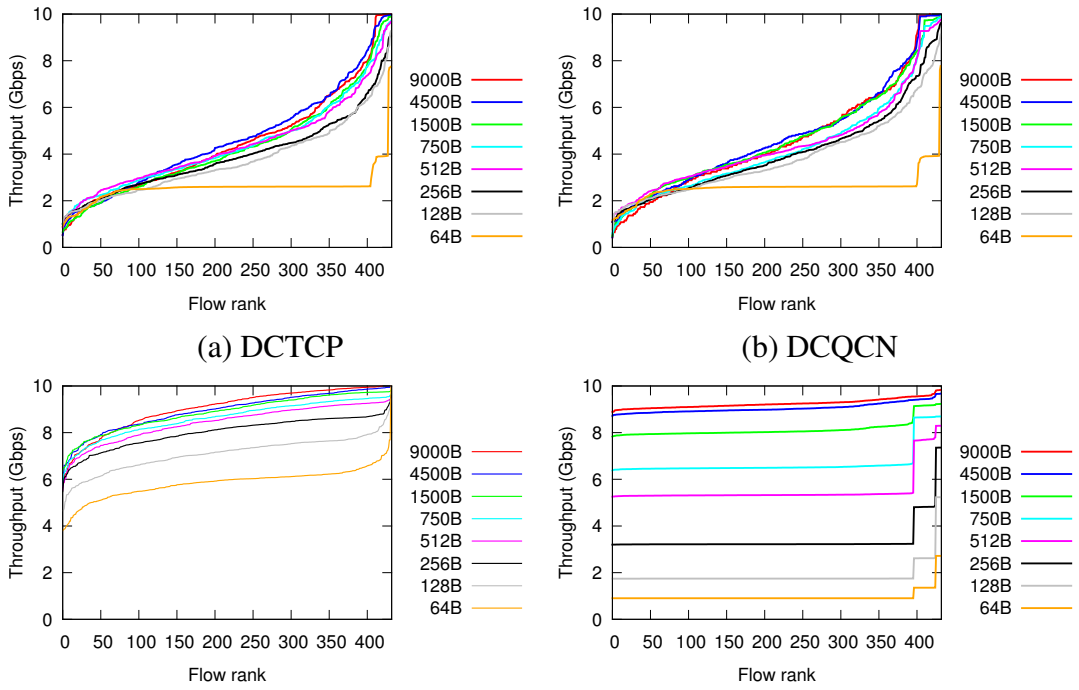
The use of a single size packet, and in particular the maximum size packet, raises the question "what is the sensitivity of the protocols to packet size?". We maintain the same experimental environment as before, but now change in each experiment the packet size. It is important to note that we do not change any parameters within the simulation environment: we want to find if the results will differ if the workload suddenly changes, not to find the optimum setup for the new workload.

Our results show that DCTCP and DCQCN are quite agnostic to packet size, with the exception of 64B which may be a corner case of the simulation environment. MPTCP is also mostly unaffected by packet sizes, though at packet sizes of 256B or less it starts to exhibit a bigger throughput loss.

In contrast to the other three protocols, NDP is extremely sensitive to packet size. The minimum and average throughput drop by approximately 14% when the packet size is changed from 9000B to 1500B. We note that 1500B is currently a common maximum transmission unit in DCN (MTU) [2, 9]. Using 750B packets will reduce the minimum and average throughput by 28% and 29.5%, respectively. Smaller packet sizes lead to even further performance loss. This is not entirely surprising, given some of the results presented in [5], e.g., in the overload paragraph in Section 6.2. The reason for the performance collapse is that NDP achieves a low compression ratio when trimming occurs. It is also expected that more packets need to be trimmed when a smaller packet size is used, thus saving even less bandwidth.

4.2 Flow completion time

We evaluate flow completion time using the fat-tree network described above, and based on flow size distributions described in [12]. Beyond reproducing the results from [5], we explore the sensitivity to flow size distribution and the repeatability of the results, both in a fully loaded setup and in an over-subscribed one. In the fully loaded setup, similar to the experiment in §4.1, there is one flow between from each node in the system, and one incoming flow. In the over-subscribed setup, there is a ratio of 4:1 of flows to nodes.



(a) DCTCP (b) DCQCN

(c) MPTCP (d) NDP

Figure 3: Per-flow throughput, permutation traffic matrix, 432-node Fat-Tree, for different protocols and packet sizes.

The setup of this simulation is slightly different to the one presented in the NDP paper, as we use the setup used in Fig. 15 ("FCT for 90KB flows with random background load, 432 node FatTree."), but with variable flow size. Our setup is consistent with the previous simulation. Each simulation runs for two (simulated) seconds, and measures the flow completion time of the simulated flows.

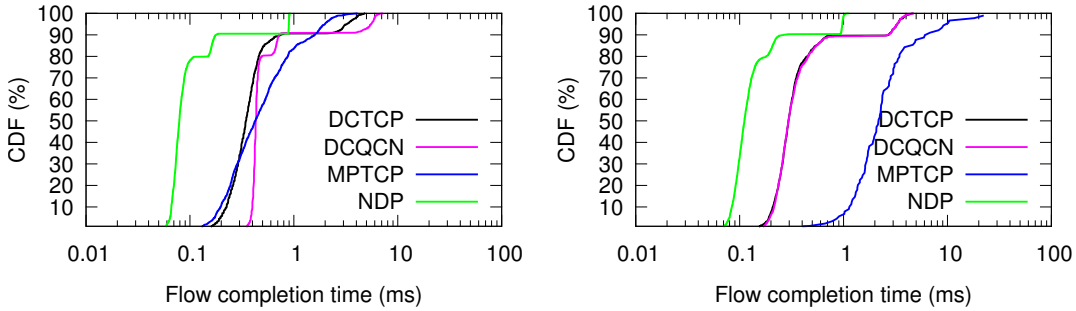
4.2.1 Workload

The workload used in the simulation is not a real one, but the flow size distribution extracted from [12]. While Roy *et al.* released traces, these traces are sampled and provide only packet size distribution. We were not able to obtain the raw data used to generate the flow size distributions presented in [12], and therefore extracted the distributions from the paper using [11] and validated manually. We use the CDFs marked "All" in Figure 6 of [12]. The distribution is in steps of 5%, with the last entry representing the 99% rather than the maximum value at the tail.

4.2.2 Results

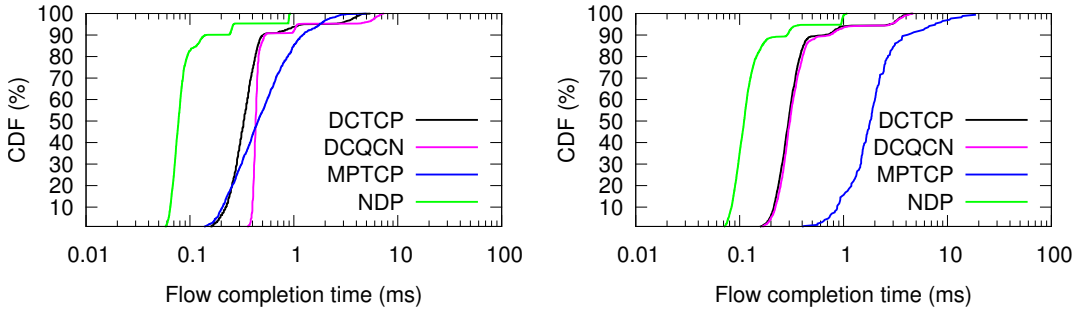
The results of our simulations largely reaffirm the claims made in [5]: The FCT of NDP outperforms other protocols both in the fully utilized (1:1) and the oversubscribed (4:1) scenarios, as shown in Figure 4, using the original distribution from the paper. The results using the extracted Web, Cache and Hadoop are presented in Figures 5,6,7, respectively. The seed used in all the experiments is 1.

An interesting point in the results above is that the minimum FCT of NDP is lower than



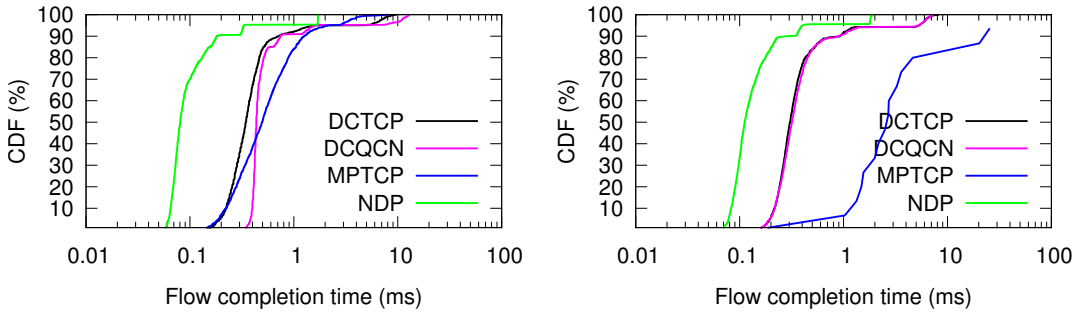
(a) 1:1 Ratio (b) 4:1 Ratio

Figure 4: FCT using the Web distribution from [5].



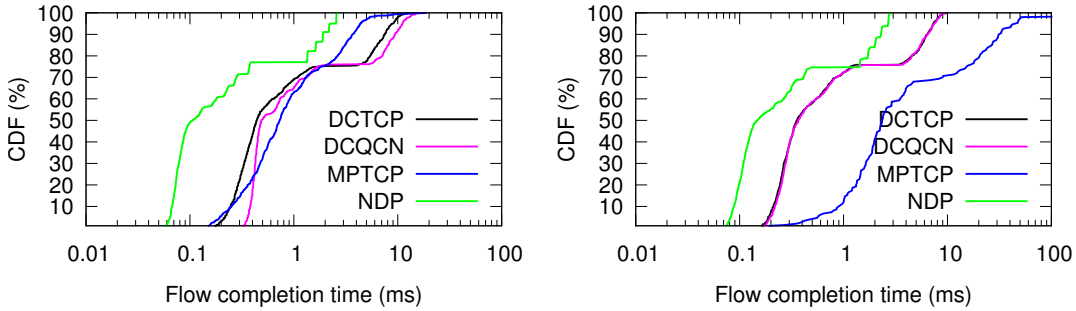
(a) 1:1 Ratio (b) 4:1 Ratio

Figure 5: FCT using the extracted Web distribution.



(a) 1:1 Ratio (b) 4:1 Ratio

Figure 6: FCT using the extracted Hadoop distribution.



(a) 1:1 Ratio (b) 4:1 Ratio

Figure 7: FCT using the extracted Cache distribution.

Percentile	Web [5]	Web	Cache	Hadoop
5%		144B	208B	288B
10%	250B	300B	326B	320B
15%		300B	360B	350B
20%	500B	375B	540B	367B
25%		550B	550B	495B
30%	1000B	670B	712B	550B
35%		1000B	1134B	600B
40%	1500B	1300B	2354B	600B
45%		1360B	7898B	615B
50%	2000B	1500B	20960B	700B
55%		1700B	55616B	770B
60%	3000B	2200B	111984B	880B
65%		2500B	174170B	1600B
70%	4000B	3500B	239778B	6680B
75%		5000B	360000B	45000B
80%	10000B	8722B	1561000B	48000B
85%		23000B	1903000B	80000B
90%	100000B	61400B	2248000B	110000B
95%		208500B	2600000B	290000B
99%	1000000B	1000000B	3067000B	2000000B

Table 1: Flow size distributions used in the simulation, extracted from [12].

Web [5], 1:1						
	Completed Flows	Min	Median	95%	99%	Maximum
NDP	1728	0.055ms	0.078ms	0.89ms	0.90ms	0.92ms
DCTCP	1235	0.097ms	0.344ms	3.20ms	4.14ms	4.8ms
DCQCN	1053	0.097ms	0.43ms	5.53ms	6.15ms	7.08ms
MPTCP	1226	0.097ms	0.42ms	2.00ms	1.65ms	4.029ms

Table 2: FCT results using Web workload, as in [5], fully utilized system.

Web [5], 4:1						
	Completed Flows	Min	Median	95%	99%	Maximum
NDP	1674	0.063ms	0.11ms	0.97ms	1.00ms	1.10ms
DCTCP	1240	0.11ms	0.29ms	3.29ms	3.92ms	4.61ms
DCQCN	1237	0.13ms	0.29ms	3.26ms	3.87ms	4.66ms
MPTCP	89	0.14ms	2.16ms	9.13ms	18.12ms	21.9ms

Table 3: FCT results using Web workload, as in [5], over-subscribed system.

Web, 1:1						
	Completed Flows	Min	Median	95%	99%	Maximum
NDP	1786	0.055ms	0.077ms	0.26ms	0.90ms	0.92ms
DCTCP	1330	0.097ms	0.33ms	1.20ms	4.20ms	5.34ms
DCQCN	1174	0.097ms	0.43ms	1.08ms	6.55ms	7.23ms
MPTCP	1239	0.11ms	0.45ms	1.77ms	2.66ms	4.89ms

Table 4: FCT results using Web workload, fully utilized system.

Web, 4:1						
	Completed Flows	Min	Median	95%	99%	Maximum
NDP	1723	0.062ms	0.11ms	0.94ms	0.99ms	1.07ms
DCTCP	1352	0.12ms	0.29ms	2.76ms	3.72ms	4.54ms
DCQCN	1332	0.13ms	0.31ms	2.79ms	3.86ms	4.59ms
MPTCP	172	0.14ms	1.79ms	6.68ms	13.63ms	18.4ms

Table 5: FCT results using Web workload, over-subscribed system.

Cache, 1:1						
	Completed Flows	Min	Median	95%	99%	Maximum
NDP	1305	0.055ms	0.11ms	2.2ms	2.57ms	2.58ms
DCTCP	643	0.15ms	0.44ms	8.73ms	10.72ms	12.6ms
DCQCN	549	0.25ms	0.48ms	10.98ms	13.47ms	15.68ms
MPTCP	827	0.13ms	0.73ms	4.32ms	8.33ms	18.44ms

Table 6: FCT results using Cache workload, fully utilized system.

Cache, 4:1						
	Completed Flows	Min	Median	95%	99%	Maximum
NDP	1215	0.062ms	0.15ms	2.68ms	2.74ms	2.8ms
DCTCP	725	0.13ms	0.35ms	7.75ms	8.57ms	9.65ms
DCQCN	720	0.13ms	0.36ms	7.44ms	8.55ms	9.37ms
MPTCP	150	0.054ms	2.35ms	40.92ms	50.95ms	831ms

Table 7: FCT results using Cache workload, over-subscribed system.

Hadoop, 1:1						
	Completed Flows	Min	Median	95%	99%	Maximum
NDP	1712	0.056ms	0.08ms	0.33ms	1.71ms	1.72ms
DCTCP	1182	0.097ms	0.34ms	1.47ms	7.73ms	9.51ms
DCQCN	1020	0.097ms	0.44ms	1.39ms	10.5ms	12.86ms
MPTCP	1167	0.13ms	0.49ms	2.06ms	4.12ms	7.86ms

Table 8: FCT results using Hadoop workload, fully utilized system.

Hadoop, 4:1						
	Completed Flows	Min	Median	95%	99%	Maximum
NDP	1663	0.062ms	0.11ms	0.40ms	1.83ms	1.89ms
DCTCP	1200	0.11ms	0.30ms	5.25ms	6.73ms	8.15ms
DCQCN	1194	0.13ms	0.32ms	5.24ms	6.75ms	7.23ms
MPTCP	15	0.14ms	2.51ms	20.28ms	—	25.23ms

Table 9: FCT results using Hadoop workload, over-subscribed system.

other protocols. We assume that this is due to NDP starting to transmit a flow as soon as a flow arrives, without a three-way handshake. A discussion of the necessity of a three-way handshake is outside the scope of this paper. To verify that, we run an experiment with a zero-load system (a single active flow), and see that the result holds. We also observe that the minimum FCT is not necessarily for the minimum flow size. More detailed results for the zero-load system appear in §4.3.

One may note that the number of flows completed by MPTCP is sometimes significantly smaller than all other protocols. Following further inspection, and while this may not be the cause, we found that in some experiments there are flows that timeout in MPTCP and never complete - there is one such flow in MPTCP cache 1:1 and MPTCP Hadoop 4:1, and eight such flows in MPTCP cache 4:1. The surprising result, however, is that we find that flows timeout also in NDP: 1 flow in NDP web 1:1, NDP web 4:1, NDP cache 1:1 and NDP Hadoop 4:1. 2 flows in NDP Hadoop 1:1 and 3 flows in NDP cache 4:1. We don't observe any NDP timeouts using the web distribution from [5]. This does, however, lead us to ask: how sensitive are the results to the seed?

4.3 Sensitivity Test

We conduct a limited sensitivity test, in which we only change the seed used in the experiment. We first focus on the Web distribution used in [5] and on NDP. Instead of running once, we run 50 times (the number of experiments is time and resources limited) and compare the results.

In the fully utilized configuration (1:1) we find two cases (seeds 26 and 38) where a single flow timeouts. In the over-subscribed scenario (4:1) we find 2 seeds (8,36) where there is a timeout. For one specific seed (10), however, the number of timeouts is no less than 153. Furthermore, the number of completed flows is just 353, compared with 1636 to 1690 in all other runs. The command used to trigger this event is:

```
<path>/htsim_ndp_perm_shortflows -o ndp_logfile4_10 -strat perm
 -nodes 432 -conns 1728 -cwnd 23 -q 8 -seed 10  > debug_ndp_4load_10
```

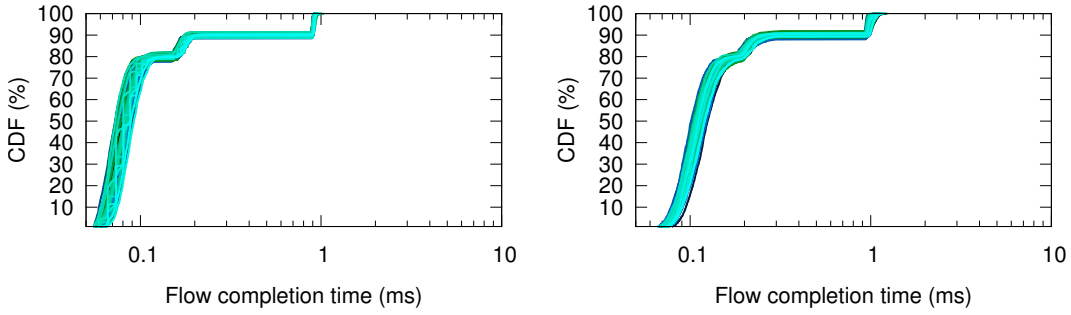
To reproduce this result, use the original NDP repository, but extend the run time in $datacenter/main\_ndp\_perm\_shortflows.cpp$ to two seconds.

We have made the authors of NDP aware of this issue, but at the time this text is written it was not resolved yet.

As a timeout was observed in MPTCP, we also explored timeouts in MPTCP for 50 different seeds, and found that in 27 out of 50 over-subscribed scenarios there were between one and four flow timeouts. We also find that the number of completed flows ranged from 2 to 1055 with the median being 371. We didn't find a correlation between flow timeouts and the number of flows completed by MPTCP.

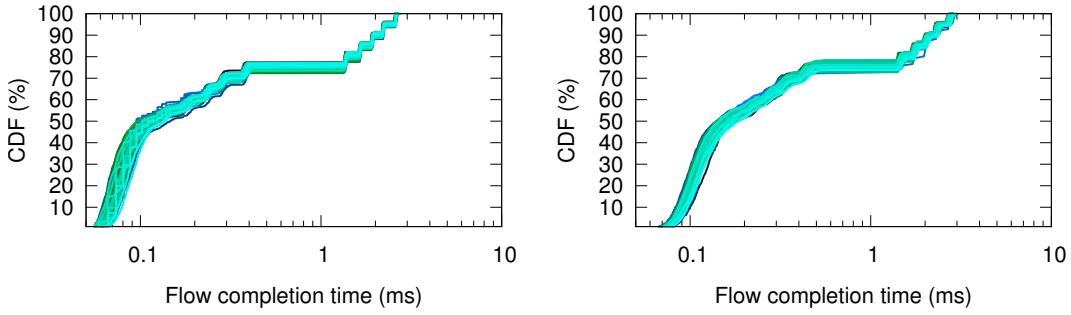
Next, we repeat the same experiment but with the Cache workload. The Cache workload is selected as NDP had the highest FCT under this workload, and we want to check whether the high FCT leads to greater performance variance under different seeds.

In the fully utilized configuration (1:1) we find 37 seeds with flow timeouts, detailed in Table 11. In the oversubscribed scenario, 40 seeds lead to timeouts, as the same table shows. The number of timeouts is significantly higher under this workload: not only in the number of seeds leading to a timeout, but also the probability for a lot of timeouts is higher, with 5 seeds leading to more than twenty timeouts. In the most extreme case, there are 176 timeouts



(a) 1:1 Ratio (b) 4:1 Ratio

Figure 8: NDP: FCT using the Web distribution used in [5], for 50 different seeds.



(a) 1:1 Ratio (b) 4:1 Ratio

Figure 9: NDP: FCT using the Cache distribution, for 50 different seeds.

(seed=22). Here, just 111 flows are completed, compared with a median of 1222. Four out of the five seeds that had more than 20 timeouts also complete less than a thousand flows. The exception is seed 46 that had 24 timeouts, where 1117 flows were completed. A different seed completed less than a thousand flows: seed 26 has 3 timeouts and completes 710 flows.

To look differently at the timeout problem, we consider the throughput, i.e., the number of bytes in completed transfers. In the fully utilized configuration (1:1) the average throughput per experiment is 780MB. In the oversubscribed scenario (4:1) it is 718MB, while for seeds where no timeouts occurs the throughput is also around 780MB. However, timeouts often lead to significant throughput loss: in experiments with tens of timeouts, the throughput drops by 20% to around 580MB, while in the worst case (seed 22) the throughput drops to 75MB, an order of magnitude less than other seeds.

When flows complete, the performance gap between them is small: the minimum FCT ranges between 55.8μs and 63.3μs (13%), and the maximum FCT ranges from 2.57ms to 2.73ms (6%)[1].

To better understand flow completion time under different scenarios, Figures 10 and 11 show the FCT per flow size, with each flow size marked by a different colour. We omit the legend from the graph for clarity, but the flow sizes correspond to Table 1, and the smallest flow has the smallest FCTs (left most lines), while the largest flow exhibits the longest FCTs (right most lines). The data used in these graphs is identical to Figures 8 and 9, only broken down by flow sizes. This breakdown of FCT by flow size explains the shape of the CDFs presented in Figures 8 and 9, as each flow size has a distinct FCT, creating "steps" shapes in the overall CDF of FCT. Small flow sizes appear as batched in Figure 10, but not in Figure 11,

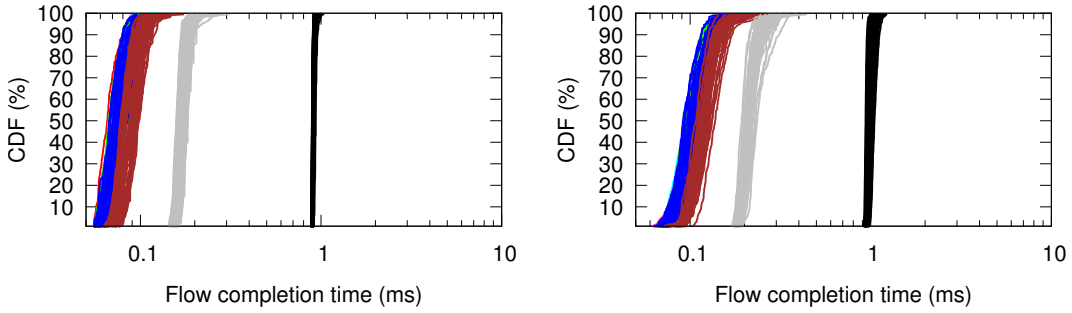
[1]The values in Table 7 are based on a different seed

Seed	#Timeouts 1:1	#Timeouts 4:1
1	1	3
2	1	0
3	0	1
4	2	3
5	1	5
6	9	36
7	7	3
8	3	0
9	8	2
10	5	0
11	4	0
12	4	0
13	1	1
14	1	1
15	0	2
16	0	1
17	1	2
18	0	2
19	3	5
20	0	0
21	0	3
22	0	176
23	2	1
24	2	43
25	5	2
26	0	3
27	0	0
28	0	0
29	4	1
30	0	6
31	1	4
32	5	0
33	3	4
34	1	45
35	5	2
36	0	1
37	4	1
38	6	2
39	0	2
40	1	5

Table 10: Cache workload: the number of timeouts occurring in each experiment.

Seed	#Timeouts 1:1	#Timeouts 4:1
41	1	4
42	3	3
43	9	3
44	1	4
45	4	3
46	1	24
47	1	0
48	1	6
49	5	2
50	2	1

Table 11: Cache workload: the number of timeouts occurring in each experiment - continued.

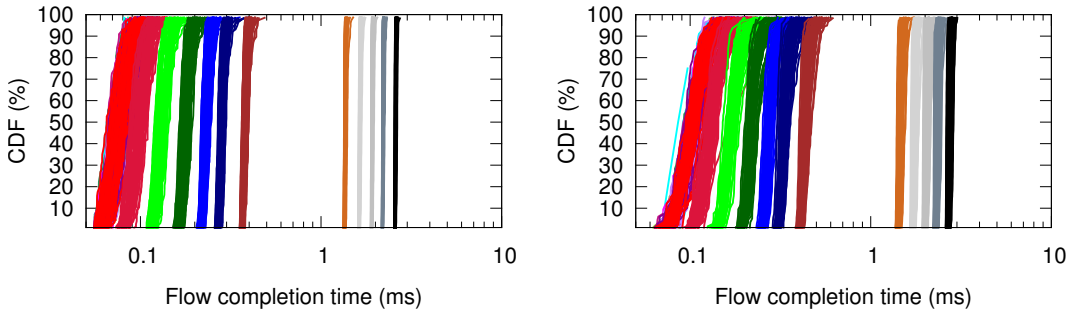


(a) 1:1 Ratio (b) 4:1 Ratio

Figure 10: NDP: FCT by flow size, using the Web distribution from [5], for 50 different seeds.

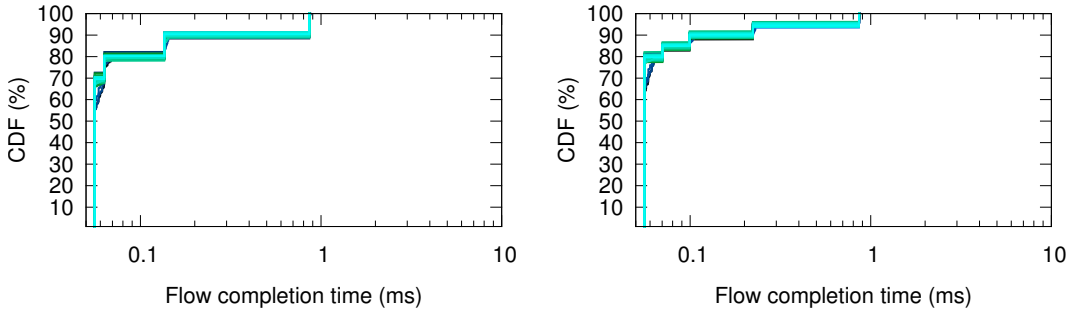
where starting 20KB flow size it becomes easy to distinguish between the FCT of different flow sizes, even in 1:1 traffic ratio scenario, due to the increased FCT.

To complete this evaluation, we study NDP's flow completion time under zero load, using fifty different seeds and for different packet size distributions. The results are presented in Figure 12, and are as one would expect: the latency is as expected, and mostly the same in all runs (some expected variations exist due to the variance in packet sizes between runs). There are no flow timeouts, and the number of flows completed is largely the same for all seeds.



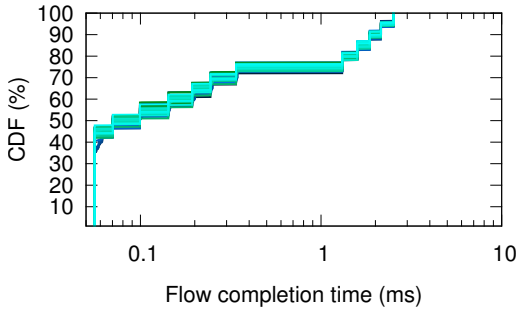
(a) 1:1 Ratio (b) 4:1 Ratio

Figure 11: NDP: FCT by flow size, using the Cache distribution, for 50 different seeds.

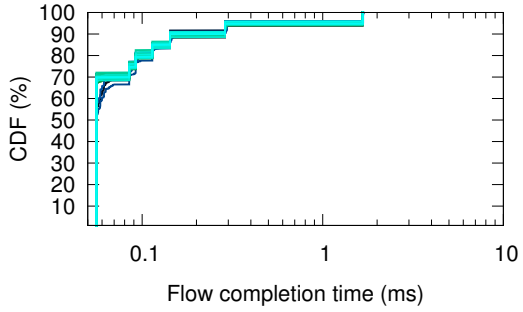


(a) Web distribution from [5].

(b) Extracted Web distribution.



(c) Cache distribution.



(d) Hadoop distribution.

Figure 12: NDP: FCT using the different distributions, using 50 different seeds, zero load.

4.4 Incast

We leave the study of incast to a later version of this paper.

4.5 Simulation Environment - Discussion

Exploring our simulations' results, we try and separate results that are a consequence of the algorithm, and results that may be caused by simulator' implementation issues. We don't have confirmed answers for any of the cases.

Per flow throughput dependence is believed to be a result of the algorithm, since the compression ratio between packet size and trimmed packet is the probable cause, and as some results in the original paper indicate packet size sensitivity. We do find it, however, concerning that there are comments in the code such as:

```
set_packet_size(9000); // it's a datacentre, use jumbograms
```

as this assumption does not match many data centre workloads [12] and configurations [2, 9], and limits the flexibility of the simulation environment (e.g., background traffic).

It is unclear if flows timeout and throughput collapse (i.e., overall throughput and completed flows per experiment) are the result of the algorithm or the simulator. With timeouts being seen also for MPTCP, it is not unlikely that the cause lies in the simulator.

The htsim simulator is modelling the network as "a collection of pipes (that add delays) and queues (with fixed processing capacity, and finite buffers)" [3]. As such, the simulator conducts simulations quite quickly, and is applicable for many functional and high-level evaluations. However, using such a model, it misses many of the intricacies of today's switches, for example, queue occupancy is only a function of packet size and not of a queue's data-bus width, which

can lead to up to 50% estimation error [15]. It is also easy to miss some of the phenomena (e.g., timeout) reported in this paper.

5 Related Work

Several works have already used NDP and evaluated it for different needs. Homa [6] compared its performance to NDP, with each solution running on its native simulator, and has shown that NDP's network utilization limit was lower than compared solutions (pias, pHost, pFabric and Homa). Homa also demonstrated that NDP shows the worst median and 99% slowdown as a function of message size. Homa reported failing to run their experiment using NDP for a network load of over 70%.

Shoal [13] also demonstrates improved performance compared to NDP, but more interestingly it shows that NDP does not performs better than DCTCP and DCQCN: it confirmed that NDP has better 99.9% FCT for short flows ($\leq 100KB$), but for different applications with disaggregated workload it had the worst 99.9% FCT. Unfortunately, Shoal uses a proprietary simulation environment that is not open source. and thus its results cannot be reproduced here.

MDTCP [8] reproduced some of NDP's results, but has shown that for other scenarios, NDP's FCT is worse than DCTCP. MDTCP used htsim for its evaluation.

Stardust [15] was the motivation for this evaluation work, and this paper extends the results provided in the NSDI'19 version of Stardust.

6 Conclusion

This paper has provided a performance evaluation of NDP, complementing the Stardust [15] paper. We report several issues that we detect while evaluating the performance, both in the hardware and the simulation environment of the protocol. In particular, we show that the hardware implementation of NDP on NetFPGA performs worse than the Reference Switch implementation on the same platform, that the simulated throughput is highly dependent on packet size, and that flow timeout can lead to significant performance drop. While the concept of NDP uses many interesting ideas, the evaluation backing these ideas, as used in the NDP paper [5], was not sufficiently rigorous. We hope that this paper will help improve the quality of NDP.

7 Acknowledgements

We would like to thank Xilinx, Cypress and Micron for their continuing support of the NetFPGA project. We thank Andrew W Moore, Changhoon Kim and Robert Soulé for their feedback on this paper. We acknowledge support by the Leverhulme Trust (ECF-2016-289) and the Isaac Newton Trust.

References

[1] G. Antichi, M. Shahbaz, Y. Geng, N. Zilberman, A. Covington, M. Bruyere, N. McKeown, N. Feamster, B. Felderman, M. Blott, et al. OSNT: Open source network tester. *IEEE Network*, 28(5):6–12, 2014.

[2] D. Firestone, A. Putnam, S. Mundkur, D. Chiou, A. Dabagh, M. Andrewartha, H. Angepat, V. Bhanu, A. Caulfield, E. Chung, et al. Azure accelerated networking: Smartnics in the public cloud. In *NSDI*, pages 51–66, 2018.

[3] M. Handley. Multipath tcp implementations. `http://nrg.cs.ucl.ac.uk/mptcp/implementation.html`.

[4] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M. Wójcik. *NDP*, 2017. Repository, `https://github.com/nets-cs-pub-ro/NDP` [accessed Dec. 2017].

[5] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M. Wójcik. Re-architecting datacenter networks and stacks for low latency and high performance. In *SIGCOMM*, pages 29–42. ACM, 2017.

[6] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout. Homa: A receiver-driven low-latency transport protocol using network priorities. In *SIGCOMM*. ACM, 2018.

[7] NetFPGA. *NetFPGA-SUME-live, issue 36: 10G port - Attachment unit - Rx side - inefficiency*, 2017. `https://github.com/NetFPGA/NetFPGA-SUME-live/issues/36`.

[8] D. B. Oljira, K.-J. Grinnemo, A. Brunstrom, and J. Taheri. MDTCP: Towards a practical multipath transport protocol for telco cloud datacenters. In *NOF*, pages 9–16, 2018.

[9] I. Pagliai. My personal Azure FAQ on Azure networking (v3), 2017. `https://blogs.msdn.microsoft.com/igorpag/2017/04/06/my-personal-azure-faq-on-azure-networking-v3/`.

[10] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving datacenter performance and robustness with multipath tcp. In *SIGCOMM Comput. Commun. Rev.*, volume 41, pages 266–277, 2011.

[11] A. Rohatgi. Webplotdigitizer, 2011. `https://automeris.io/WebPlotDigitizer`.

[12] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the social network's (datacenter) network. In *SIGCOMM Comput. Commun. Rev.*, volume 45, pages 123–137. ACM, 2015.

[13] V. Shrivastav, A. Valadarsky, H. Ballani, P. Costa, K. S. Lee, H. Wang, R. Agarwal, and H. Weatherspoon. Shoal: A network architecture for disaggregated racks. In *NSDI*, 2019.

[14] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore. NetFPGA SUME: Toward 100 Gbps as research commodity. *IEEE MICRO*, 34(5):32–41, Sept. 2014.

[15] N. Zilberman, G. Bracha, and G. Schzukin. Stardust: Divide and conquer in the data center network. In *NSDI*, pages 141–160, 2019.