Performance Implications of Problem Decomposition Approaches for SDN Pipelines

William Brockelsby
NC State University Department of Computer Science
Duke University Office of Information Technology
Email: wjbrocke@ncsu.edu

Rudra Dutta
NC State University
Department of Computer Science
Email: rdutta@ncsu.edu

***Abstract*—Software defined networking (SDN) allows organizations to modify networks programmatically to implement custom forwarding behavior and to react to changing conditions. While there are many approaches available to implement SDN those that leverage forwarding table abstractions such as OpenFlow and P4 require developers to decompose problems into one or more tables associated with a definable pipeline. This paper explores tradeoffs between table depth and pipeline length associated with different problem decomposition options by analyzing the performance impact on hardware and software data planes including software data planes leveraging hardware acceleration through the use of SmartNICs.**

***Index Terms*—computer networks, software defined networking**

1. Introduction

One approach to Software Defined Networking (SDN) involves the separation of the control and forwarding planes through the use of an abstraction as popularized by OpenFlow [1]. Initially the abstraction used with OpenFlow modeled the forwarding plane as a single table consisting of columns representing fields in frame/packet headers and rows representing match criteria [1]. Utilizing a single table provides a simple abstraction for developers but often leads to a large number of table entries due to the resultant Cartesian product associated with mapping multiple sets of attributes onto a single table [2]. Newer versions of OpenFlow [3] and later P4 [4] introduced a pipeline model which chains multiple tables together by utilizing goto actions in support of efficient resource utilization.

The primary drawback associated with the multi-table approach is that it can be difficult to realize an arbitrary multi-table pipeline on traditional fixed function ASIC based switching platforms [2]. In this paper we explore the tradeoffs between table depth and pipeline length to determine any performance differences across both hardware and software data planes in terms of latency and packet loss at various throughput rates. The results of this analysis will highlight the importance of analyzing data plane characteristics while decomposing applications into an appropriately structured pipeline in order to maximize resource utilization while delivering the best performance to the application.

2. Motivation

To illustrate a few pipeline decomposition approaches, we provide an example in which a data plane is utilized to implement Access Control Lists (ACLs) capable of matching IP addresses, transport layer protocols and port numbers as well layer-2 headers in support of switching. In order to facilitate comparison of decomposition options, we indicate the amount of resources consumed by calculating the product of the number of rows and match fields (columns) required by the approach using sample data. The first decomposition shown in Pipeline 1 illustrates the single-table approach which consumes 200 match entries. An alternate decomposition is shown in Pipeline 2 which decomposes the problem into two tables – with Table 0 dedicated to ACL functionality and Table 1 dedicated to the Layer 2 forwarding consumes only 54 entries.

SINGLE-TABLE(#0)

VL	ET	DMAC	SNET	DNET	PRO	TD	UD	ACT
1	IP	M2	J/24	K/24	TCP	80	*	VL=11,OUT=2
1	IP	M2	J/24	K/24	TCP	443	*	VL=11,OUT=2
1	IP	M4	K/24	L/24	TCP	3306	*	VL=11,OUT=2
1	IP	M4	K/24	L/24	UDP	*	53	VL=11,OUT=2
1	IP	M4	K/24	L/24	UDP	*	123	VL=11,OUT=2
1	IP	M4	K/24	L/24	UDP	*	514	VL=11,OUT=2
2	IP	M6	J/24	K/24	TCP	80	*	VL=12,OUT=2
2	IP	M6	J/24	K/24	TCP	443	*	VL=12,OUT=2
2	IP	M8	K/24	L/24	TCP	3306	*	VL=12,OUT=2
2	IP	M8	K/24	L/24	UDP	*	53	VL=12,OUT=2
2	IP	M8	K/24	L/24	UDP	*	123	VL=12,OUT=2
2	IP	M8	K/24	L/24	UDP	*	514	VL=12,OUT=2
1	IP	*	J/24	K/24	TCP	80	*	VL=11,GRP=2
1	IP	*	J/24	K/24	TCP	443	*	VL=11,GRP=1
1	IP	*	K/24	L/24	TCP	3306	*	VL=11,GRP=1
1	IP	*	K/24	L/24	UDP	*	53	VL=11,GRP=1
1	IP	*	K/24	L/24	UDP	*	123	VL=11,GRP=1
1	IP	*	K/24	L/24	UDP	*	514	VL=11,GRP=1
2	IP	*	J/24	K/24	TCP	80	*	VL=12,GRP=2
2	IP	*	J/24	K/24	TCP	443	*	VL=12,GRP=2
2	IP	*	K/24	L/24	TCP	3306	*	VL=12,GRP=2
2	IP	*	K/24	L/24	UDP	*	53	VL=12,GRP=2
2	IP	*	K/24	L/24	UDP	*	123	VL=12,GRP=2
2	IP	*	K/24	L/24	UDP	*	514	VL=12,GRP=2
*	*	*	*	*	*	*	*	DROP

Table #0 Size: 25 rows x 8 match fields = 200

Pipeline 1: Single-Table decomposition

The third decomposition option shown in Pipeline 3 splits the ACL functionality across three discrete tables and continues to utilize the Layer 2 forwarding table consuming in total only 53 entries.

ACL-TABLE(#0)

ET	SNET	DNET	PRO	TD	UD	ACT
IP	J/24	K/24	TCP	80	*	GOTO=1
IP	J/24	K/24	TCP	443	*	GOTO=1
IP	K/24	L/24	TCP	3306	*	GOTO=1
IP	K/24	L/24	UDP	*	53	GOTO=1
IP	K/24	L/24	UDP	*	123	GOTO=1
IP	K/24	L/24	UDP	*	514	GOTO=1
*	*	*	*	*	*	DROP

Table #0 Size: 7 rows x 6 match fields = 42

L2-TABLE(#1)

VL	DMAC	ACT
1	M2	VL=11,OUT=2
1	M4	VL=11,OUT=2
2	M6	VL=12,OUT=2
2	M8	VL=12,OUT=2
1	*	VL=11,GRP=1
2	*	VL=12,GRP=2

Table #1 Size: 6 rows x 2 match fields = 12

Total Size: $\sum(|T0|, |T1|) = 42 + 12 = 54$

Pipeline 2: Two-Table Decomposition

ACL-DIVIDE TABLE(#0)

ET	SNET	DNET	ACT
IP	J/24	K/24	GOTO=1
IP	K/24	L/24	GOTO=2
*	*	*	DROP

Table #0 Size: 3 rows x 3 match fields = 9

L4-CLIENT-TRANSPORT TABLE(#1)

ET	PRO	TD	UD	ACT
IP	TCP	80	*	GOTO=3
IP	TCP	443	*	GOTO=3
*	*	*	*	DROP

Table #1 Size: 3 rows x 4 match fields = 12

L4-SERVER-TRANSPORT TABLE(#2)

ET	PRO	TD	UD	ACT
IP	TCP	3306	*	GOTO=3
IP	UDP	*	53	GOTO=3
IP	UDP	*	514	GOTO=3
IP	UDP	*	123	GOTO=3
*	*	*	*	DROP

Table #2 Size: 5 rows x 4 match fields = 20

L2 TABLE(#3)

VL	DMAC	ACT
1	M2	VL=11,OUT=2
1	M4	VL=11,OUT=2
2	M6	VL=12,OUT=2
2	M8	VL=12,OUT=2
1	*	VL=11,GRP=1
2	*	VL=12,GRP=2

Table #3 Size: 6 rows x 2 match fields = 12

Total Size: $\sum(|T0|, |T1|, |T2|, |T3|) = 9 + 12 + 20 + 12 = 53$

Pipeline 3: Four-Table Decomposition

3. Approach

In order to evaluate the tradeoff between table depth and pipeline length when decomposing a problem we measure the latency and loss associated with each SDN data plane with different pipeline configurations utilizing a Spirent TestCenter as illustrated in Fig. 3.1. The experiment is conducted on both hardware and software data planes to assess whether or not the results vary depending on the underlying characteristics of the data plane. We utilize the NS2122 [5] hardware data plane from NoviFlow which leverages a Mellanox NP-5 network processor and an Edge-Core Wedge 100BF-32X [6] which leverages a Barefoot Networks Tofino SDN ASIC. We also leverage Open vSwitch (OVS) [7] version 2.11.0 on CentOS 7.7 Linux hosts utilizing Intel Xeon Silver 4110 (8-Core 2.1GHz, 11M Cache) processors with 64GB of RAM to evaluate a variety of NICs including a traditional Intel X710-DA2 2-port 10Gb/s NIC as well as 10Gb/s [8], 25Gb/s [9] and 40Gb/s [10] NICs from Netronome which are capable of operating with standard NIC firmware or with SmartNIC firmware which accelerates OVS in hardware by leveraging an on-NIC network processor. All of the test cases utilize the Ryu [11] 4.32 OpenFlow controller running on Python [12] 3.6.8 to populate the flow tables within the data planes under test.

To explore the tradeoff between table depth and pipeline length we measure data plane latency and packet loss under different traffic rates (0.25, 0.5, 1, 2.5, 5 and 9 Gb/s) and optionally (24, 39, 90 and 99 Gb/s) on platforms supporting high-speed interfaces utilizing uni-directional 128-byte frames that ingress the first port of the device and egress the second port. We conduct the following test cases across each data plane:

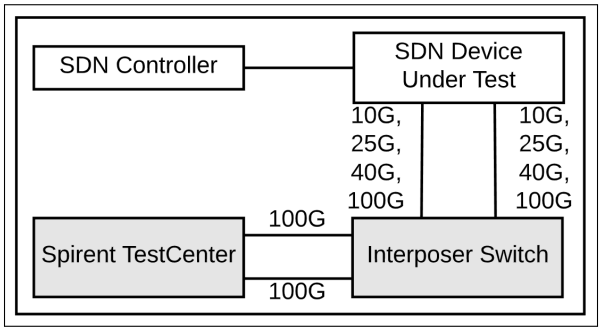

Fig. 3.1: Test Environment

Test Case 1 – Control: In order to perform a baseline control test case for each data plane, we deploy a single-table pipeline with a simple rule instructing data ingressing one port to egress a second port and vice-versa. The match associated with these flow table entries utilize wildcards for all protocol headers and only matches the ingress port.

Test Case 2 – Table Depth: This test case involves analyzing the general performance impact of table depth. For each data plane we deploy a single-table pipeline with 1,024 high priority flow table entries that match on unique MAC addresses not found within the primary test stream.

As a result, traffic associated with the primary test stream will only match the low priority flow table entry at the end of the table. We then repeat the same test scenario but instead with 32,768 high priority flow table entries that match on unique MAC addresses not found within the primary test stream. Note that in both cases, we do introduce a separate 1Mb/s test stream spread across the 1,024 or 32,768 high priority entries for the purposes of keeping the high priority entries active within the data plane to ensure they are not aged out due to any data plane flow cache optimizations.

Test Case 3 – Pipeline Length: This test case analyzes the performance impact of pipeline length by chaining tables together to form pipelines of length N = 2, 4, 8 where tables I in 0..N-2 contain a single goto statement to table I+1 and table N-1 contains a simple rule instructing data ingressing one port to egress the other port and vice versa. The match associated with these flow table entries utilize wildcards for all protocol headers and only match the ingress port.

Test Cases 4-6 involve constructing the pipelines associated with the sample decompositions described in §2 shown within Pipelines 1-3 to analyze the performance of various decomposition scenarios.

4. Results

The results of the Test Cases (TC) listed in §3 are presented in tabular format for each data plane within this section. The column heading convention used in the remaining tables and discussion is presented within Table 4.1 for reference.

Table 4.1: Table Column Heading Descriptions

Label	Description
Gb/s	Layer-1 traffic rate presented to device under test
P	Number of tables in the pipeline for pipeline length tests
NS2122	NoviFlow NoviSwitch model NS2122
WB5132	Edge-Core 100BF-32X switch with NoviFlow NOS
IS:10G	Intel X710-DA2 standard 2x10G NIC
NS:10G	Netronome Agilio CX 2x10G NIC - standard firmware
NO:10G	Netronome Agilio CX 2x10G NIC - offload firmware
NS:25G	Netronome Agilio CX 2x25G NIC - standard firmware
NO:10G	Netronome Agilio CX 2x25G NIC - offload firmware
NS:40G	Netronome Agilio CX 2x40G NIC - standard firmware
NO:40G	Netronome Agilio CX 2x40G NIC - offload firmware

For each test case, tables with (HW/Offload) in the caption present results for the purpose-built hardware data planes (NS2122, WB5132) and SmartNICs utilizing tc-flower [13] offload firmware. For comparison, we also present results for a traditional NIC and non-offloaded SmartNICs utilizing standard firmware for TC1 within tables captioned as (Standard) note that these results are omitted for other test cases due to space constraints. All results are collected at the specified layer-1 data rate utilizing uni-directional 128-byte frames as described in §3 unless otherwise specified. Cases with traffic loss are identified by *italicized* latency or loss percentages within the result tables.

Table 4.2: TC1/Control (HW/Offload) – Avg Latency (µs)

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	11.28	7.22	11.84	10.89	10.74
0.50	15.27	7.22	11.84	10.89	10.74
1.00	14.20	7.22	11.84	10.89	10.75
2.50	17.50	7.22	11.85	10.91	10.76
5.00	14.02	7.22	11.93	11.13	10.96
9.00	12.58	7.23	12.34	11.06	10.94
24.00	12.51	7.22		14.26	13.57
39.00	11.28	7.23			*165.22*
90.00	11.31	7.23			
99.00	11.49	7.24			

Table 4.3: TC1/Control (Standard) – Avg Latency (µs)

Gb/s	IS:X710	NS:10G	NS:25G	NS:40G
0.25	69.55	50.57	45.92	44.42
0.50	53.70	46.64	39.33	38.40
1.00	22.43	*2,001.16*	*1,323.42*	*2,900.90*
2.50	*468.56*	*3,391.56*	*4,795.54*	*4,800.69*
5.00	*481.43*	*4,716.60*	*4,619.57*	*4,847.44*
9.00	*492.48*	*4,815.57*	*4,839.42*	*4,940.03*
24.00			*4,835.51*	*4,799.73*
39.00				*4,960.97*

TC1 control results are shown in Tables 4.2 - 4.5. Tables 4.2 and 4.4 highlight that no traffic loss or significant latency occurs on any test case associated with the purpose-built SDN switches or SmartNICs with tc-flower offload firmware except for with NO:40G under high load at 39Gb/s with a small 128-Byte frame size. The trend with NO:40G and loss at 39Gb/s with 128-Byte frames continues across other test cases and is possibly a hardware limitation as this NIC utilizes the same NFP-4000 network processor as the other SmartNICs [14] and can achieve line rate performance with a larger frame size. Another interesting, but expected, observation is the variation in latency across the network processor based platforms such as NS2122 and the SmartNICs as compared to the WB5132 Tofino ASIC platform which provides consistently lower latency across many data rates. The significant loss and latency of the standard NIC and SmartNICs with traditional non-offloaded firmware highlight the benefit of offload-capable hardware when employed in server-based data planes.

TC2 table depth results are shown in Tables 4.6 - 4.9 for both the 1,024+2 row and 32,768+2 row scenarios. Tables 4.6 - 4.9 highlight no traffic loss or significant latency on any test case associated with the purpose-built SDN switches or

Table 4.4: TC1/Control (HW/Offload) – Loss %

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	0.00	0.00	0.00	0.00	0.00
0.50	0.00	0.00	0.00	0.00	0.00
1.00	0.00	0.00	0.00	0.00	0.00
2.50	0.00	0.00	0.00	0.00	0.00
5.00	0.00	0.00	0.00	0.00	0.00
9.00	0.00	0.00	0.00	0.00	0.00
24.00	0.00	0.00		0.00	0.00
39.00	0.00	0.00			*19.25*
90.00	0.00	0.00			
99.00	0.00	0.00			

Table 4.5: TC1/Control (Standard) – Loss %

Gb/s	IS:X710	NS:10G	NS:25G	NS:40G
0.25	0.00	0.00	0.00	0.00
0.50	0.00	0.00	0.00	0.00
1.00	0.00	*1.38*	*0.85*	*2.12*
2.50	*50.52*	*42.65*	*59.69*	*59.73*
5.00	*76.29*	*79.48*	*79.06*	*80.04*
9.00	*86.47*	*88.83*	*88.89*	*89.12*
24.00			*95.83*	*95.80*
39.00				*97.43*

Table 4.6: TC2/1K Depth (HW/Offload) – Avg Latency (µs)

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	11.27	7.22	11.83	10.88	10.74
0.50	15.26	7.22	11.83	10.88	10.74
1.00	14.18	7.23	11.83	10.88	10.74
2.50	17.49	7.23	11.84	10.9	10.78
5.00	14.01	7.23	11.85	11.07	10.89
9.00	12.57	7.22	12.1	11.07	10.95
24.00	12.5	7.23		14.16	13.47
39.00	11.26	7.22			*106.49*
90.00	11.3	7.23			
99.00	11.48	7.24			

SmartNICs with tc-flower offload firmware except for the previously mentioned NO:40G at 39Gb/s. While running these experiments, we also collected the 1-minute Linux system load average for OVS test cases and noted that the system load was consistently near 0.0 for the SmartNICs operating with offload firmware but did note a 1-minute load of up to 1.25 under the 32,768+2 scenario caused by OVS flow statistics collection and maintenance and not due to host-based packet forwarding.

TC3 pipeline length results are shown in Tables 4.10 - 4.11 for pipelines of length 2, 4, 8. Both tables highlight an interesting observation not seen in the prior test cases:

Table 4.7: TC2/1K Depth (HW/Offload) – Loss %

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	0.00	0.00	0.00	0.00	0.00
0.50	0.00	0.00	0.00	0.00	0.00
1.00	0.00	0.00	0.00	0.00	0.00
2.50	0.00	0.00	0.00	0.00	0.00
5.00	0.00	0.00	0.00	0.00	0.00
9.00	0.00	0.00	0.00	0.00	0.00
24.00	0.00	0.00		0.00	0.00
39.00	0.00	0.00			*19.01*
90.00	0.00	0.00			
99.00	0.00	0.00			

Table 4.8: TC2/32K Depth (HW/Offload) – Avg Latency (µs)

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	11.25	7.23	12.25	11.23	11.01
0.50	14.16	7.23	12.25	11.23	11.08
1.00	15.24	7.23	12.25	11.23	10.76
2.50	16.14	7.23	12.32	11.26	10.81
5.00	17.48	7.23	12.48	11.36	10.96
9.00	12.56	7.22	13.04	11.58	10.94
24.00	12.48	7.23		14.28	13.54
39.00	11.26	7.23			*164.93*
90.00	11.29	7.24			
99.00	14.43	7.24			

Table 4.9: TC2/32K Depth (HW/Offload) – Loss %

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	0.00	0.00	0.00	0.00	0.00
0.50	0.00	0.00	0.00	0.00	0.00
1.00	0.00	0.00	0.00	0.00	0.00
2.50	0.00	0.00	0.00	0.00	0.00
5.00	0.00	0.00	0.00	0.00	0.00
9.00	0.00	0.00	0.00	0.00	0.00
24.00	0.00	0.00		0.00	0.00
39.00	0.00	0.00			*19.14*
90.00	0.00	0.00			
99.00	0.00	0.00			

Table 4.10: TC3/PLength (HW/Offload) – Avg Latency (µs)

P	Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
2	0.25	13.52	8.12	11.86	10.9	10.77
2	0.50	16.44	8.12	11.85	10.91	10.77
2	1.00	17.51	8.12	11.85	10.91	10.77
2	2.50	19.57	8.12	11.86	10.95	10.78
2	5.00	16.07	8.12	11.89	11.16	10.95
2	9.00	14.62	8.12	12.48	11.18	10.98
2	24.00	17.33	8.12		14.26	13.61
2	39.00	13.35	8.12			*165.3*
2	90.00	13.38	8.12			
2	99.00	13.57	8.13			
4	0.25	17.77	9.68	11.86	10.9	10.77
4	0.50	21.52	9.68	11.86	10.91	10.77
4	1.00	20.48	9.68	11.86	10.91	10.77
4	2.50	23.56	9.69	11.88	10.92	10.78
4	5.00	19.99	9.68	11.9	11.17	10.94
4	9.00	18.56	9.69	12.3	11.18	11.08
4	24.00	18.49	9.7		14.37	13.66
4	39.00	17.29	9.69			*165.5*
4	90.00	*90.11*	9.76			
4	99.00	*90.28*	9.84			
8	0.25	25.69	12.82	11.85	10.9	10.77
8	0.50	29.42	12.82	11.85	10.91	10.77
8	1.00	28.38	12.82	11.86	10.91	10.78
8	2.50	31.48	12.82	11.88	10.96	10.8
8	5.00	27.81	12.82	11.92	11.18	10.97
8	9.00	26.28	12.84	12.41	11.19	11.09
8	24.00	*185.99*	12.85		14.39	13.61
8	39.00	*193.14*	12.96			*165.28*
8	90.00	*194.38*	*115.08*			
8	99.00	*194.46*	*180.26*			

significant loss and latency is observed at higher traffic rates of pipeline length 4 and 8 on NS2122 and also with pipeline length 8 with WB5132. Interestingly, with the OVS SmartNIC solution utilizing tc-flower offload, we do not observe this behavior with the exception of the consistent loss at 39Gb/s for NO:40G as described previously. This observation is analyzed in further detail within §5.

TC4 and TC5 represent sample decomposition Pipelines 1 and 2, respectively, and have results similar to those of TC1 and TC2 with the only significant latency/loss at 39Gb/s for NO:40G as described previously. TC6 represents sample decomposition Pipeline 3 which leverages a 4-table pipeline and we observe high latency and loss with NS2122 for 90Gb/s and 99Gb/s which were also observed in TC3 although they were more severe in TC3 than in TC6.

Table 4.11: TC3/PLength (HW/Offload) – Loss %

P	Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
2	0.25	0.00	0.00	0.00	0.00	0.00
2	0.50	0.00	0.00	0.00	0.00	0.00
2	1.00	0.00	0.00	0.00	0.00	0.00
2	2.50	0.00	0.00	0.00	0.00	0.00
2	5.00	0.00	0.00	0.00	0.00	0.00
2	9.00	0.00	0.00	0.00	0.00	0.00
2	24.00	0.00	0.00		0.00	0.00
2	39.00	0.00	0.00			*19.27*
2	90.00	0.00	0.00			
2	99.00	0.00	0.00			
4	0.25	0.00	0.00	0.00	0.00	0.00
4	0.50	0.00	0.00	0.00	0.00	0.00
4	1.00	0.00	0.00	0.00	0.00	0.00
4	2.50	0.00	0.00	0.00	0.00	0.00
4	5.00	0.00	0.00	0.00	0.00	0.00
4	9.00	0.00	0.00	0.00	0.00	0.00
4	24.00	0.00	0.00		0.00	0.00
4	39.00	0.00	0.00			*19.35*
4	90.00	*43.53*	0.00			
4	99.00	*48.79*	0.00			
8	0.25	0.00	0.00	0.00	0.00	0.00
8	0.50	0.00	0.00	0.00	0.00	0.00
8	1.00	0.00	0.00	0.00	0.00	0.00
8	2.50	0.00	0.00	0.00	0.00	0.00
8	5.00	0.00	0.00	0.00	0.00	0.00
8	9.00	0.00	0.00	0.00	0.00	0.00
8	24.00	*6.20*	0.00		0.00	0.00
8	39.00	*42.27*	0.00			*19.26*
8	90.00	*74.99*	*81.46*			
8	99.00	*77.25*	*86.54*			

Table 4.12: TC4/Pipeline1 (HW/Offload) – Avg Latency (µs)

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	11.51	7.34	12.36	11.31	11.16
0.50	15.5	7.34	12.35	11.31	11.16
1.00	14.44	7.34	12.36	11.31	11.16
2.50	17.71	7.33	12.37	11.32	11.16
5.00	14.22	7.34	12.4	11.36	11.18
9.00	12.73	7.33	13.25	11.48	11.33
24.00	12.8	7.33		16.86	15.42
39.00	11.49	7.33			*170.46*
90.00	11.52	7.34			
99.00	11.67	7.35			

5. Analysis and Insights

Latency increased with pipeline length in the purpose-built hardware data planes NS2122 and WB5132 ultimately leading to loss when long pipelines were subject to high traffic rates. While it is difficult to analyze vendor-proprietary hardware implementation details that may lead to this situation, intuitively, we understand that an increase in pipeline length causes the frame to stay within the forwarding mechanism for a longer period of time ultimately leading to loss. Interestingly this is not observed with the OVS data planes as noted in §4 and is due to an implementation strategy used within OVS: While OVS exposes a multi-table pipeline abstraction via the OpenFlow to the SDN controller, under the hood, it realizes that multi-table pipeline as a single large "megaflow" classifier [15]. Regarding NS2122, note that loss and high latency were observed both in TC3 and in TC6 but the impact was

Table 4.13: TC4/Pipeline1 (HW/Offload) – Loss %

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	0.00	0.00	0.00	0.00	0.00
0.50	0.00	0.00	0.00	0.00	0.00
1.00	0.00	0.00	0.00	0.00	0.00
2.50	0.00	0.00	0.00	0.00	0.00
5.00	0.00	0.00	0.00	0.00	0.00
9.00	0.00	0.00	0.00	0.00	0.00
24.00	0.00	0.00		0.00	0.00
39.00	0.00	0.00			*21.77*
90.00	0.00	0.00			
99.00	0.00	0.00			

Table 4.14: TC5/Pipeline2 (HW/Offload) – Avg Latency (µs)

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	13.61	8.12	12.35	11.09	11.17
0.50	17.62	8.12	14.21	11.31	11.17
1.00	16.6	8.12	12.36	11.31	11.17
2.50	17.79	8.12	12.37	11.32	11.19
5.00	16.18	8.13	12.41	11.35	11.21
9.00	14.73	8.12	13.26	11.48	11.37
24.00	13.38	8.12		16.86	15.43
39.00	13.4	8.12			*170.46*
90.00	13.43	8.12			
99.00	13.63	8.15			

more severe within TC3 than TC6 even though both TC3 and TC6 utilize a 4-table pipeline. The disparity is caused by the fact that frames pass through all tables within TC3 but only 3/4 of the tables within TC6. Based on these observations, we have developed the following procedure tha can be utilized while analyzing decomposition options for SDN data planes:

ANALYSIS PROCEDURE:

- Attach the data plane under test to a traffic generator/receiver in a lab environment. For the specified/required traffic rate and frame size, record whether loss is detected. Successively increase pipeline length by using a set of daisy-chained tables with goto statements where the final table contains a match required for traffic forwarding between the generator/receiver. Denote the pipeline length where loss occurs as N.
- For the proposed application, analyze the pipeline length, L, used by the selected decomposition approach.
  - If $L < N$, based on our static analysis, no loss should be encountered unless other conditions are contributors
  - If$L >= N$, loss is is possible but only if the set of successively chained tables via goto is $>= N$. This condition is difficult to statically analyze, but can be monitored at runtime through observation of the longest chain of pipelined tables instantiated. The controller could be configured to log/alert on this condition if/when encountered.

Table 4.15: TC5/Pipeline2 (HW/Offload) – Loss %

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	0.00	0.00	0.00	0.00	0.00
0.50	0.00	0.00	0.00	0.00	0.00
1.00	0.00	0.00	0.00	0.00	0.00
2.50	0.00	0.00	0.00	0.00	0.00
5.00	0.00	0.00	0.00	0.00	0.00
9.00	0.00	0.00	0.00	0.00	0.00
24.00	0.00	0.00		0.00	0.00
39.00	0.00	0.00			*21.72*
90.00	0.00	0.00			
99.00	0.00	0.00			

Table 4.16: TC6/Pipeline3 (HW/Offload) – Avg Latency (µs)

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	16.21	8.9	12.36	11.31	11.17
0.50	20.19	8.91	12.36	11.31	11.17
1.00	19.24	8.91	12.36	11.31	11.17
2.50	22.25	8.9	12.37	11.32	11.17
5.00	18.9	8.9	12.39	11.35	11.19
9.00	17.42	8.91	13.26	11.49	11.33
24.00	17.4	8.9		16.86	15.43
39.00	16.07	8.91			*170.45*
90.00	*62.86*	8.93			
99.00	*63.19*	8.96			

Table 4.17: TC6/Pipeline3 (HW/Offload) – Loss %

Gb/s	NS2122	WB5132	NO:10G	NO:25G	NO:40G
0.25	0.00	0.00	0.00	0.00	0.00
0.50	0.00	0.00	0.00	0.00	0.00
1.00	0.00	0.00	0.00	0.00	0.00
2.50	0.00	0.00	0.00	0.00	0.00
5.00	0.00	0.00	0.00	0.00	0.00
9.00	0.00	0.00	0.00	0.00	0.00
24.00	0.00	0.00		0.00	0.00
39.00	0.00	0.00			*21.75*
90.00	*15.88*	0.00			
99.00	*23.52*	0.00			

While prior work exists towards the development of algorithms to assist with the construction of SDN pipelines [16] we have shown that maximal decomposition may impact latency and ultimately lead to packet loss even if the pipeline is more efficient with regard to memory consumption. We have highlighted the value in conducting real-world analysis and experimentation of proposed SDN data planes prior to deployment and have developed an analysis procedure to help determine the maximum pipeline length for the anticipated traffic traffic rate and frame size. We then used this length to highlight both static and dynamic analysis procedures that can be used to inform application decomposition onto the SDN data plane to ensure maximum performance while simultaneously conserving memory resources through the deployment of a multi-table pipeline. Our testing also highlights a significant increase in capacity, scale and performance of modern SDN data planes that was unheard of only a few years ago and also illustrated advancements in non-traditional server-based forwarding engines. Given that all of the platforms we tested supported significantly deeper tables than prior-generation data planes and that there were almost no performance impacts associated with the deep table deployment scenarios we have shown that the SDN application developer may want to consider a careful balance between table depth and pipeline length when selecting a decomposition strategy for specific application scenarios.

References

[1] *OpenFlow Switch Specification Version 1.0.0*, Dec 2009. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf

[2] O. N. Foundation, "The benefits of multiple flow tables and ttps," ONF Technical Report, Tech. Rep., 2015. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/

[3] *OpenFlow Switch Specification Version 1.1.0*, Feb 2011. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.1.0.pdf

[4] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[5] *NoviFlow NS2122 Data Sheet*, Nov 2019. [Online]. Available: https://noviflow.com/wp-content/uploads/2019/11/NoviSwitch-2122-Datasheet-400_V5.pdf

[6] *Edge-Core Wedge 100BF-32X Data Sheet*, Dec 2019. [Online]. Available: https://www.edge-core.com/_upload/images/Wedge100BF-32X_65X_DS_R05_20191210.pdf

[7] *Open Virtual Switch (OVS)*. [Online]. Available: http://openvswitch.org/

[8] *Netronome Agilio CX 2x10G Data Sheet*, Jul 2018. [Online]. Available: https://www.netronome.com/m/documents/PB_Agilio_CX_2x10GbE.pdf

[9] *Netronome Agilio CX 2x25G Data Sheet*, Jul 2018. [Online]. Available: https://www.netronome.com/m/documents/PB_Agilio_CX_2x25GbE.pdf

[10] *Netronome Agilio CX 2x40G Data Sheet*, Mar 2017. [Online]. Available: https://www.netronome.com/m/documents/PB_Agilio_CX_2x40GbE.pdf

[11] S. Ryu, "Framework community: Ryu sdn framework," *Online. http://osrg. github. io/ryu*, 2015.

[12] G. Van Rossum *et al.*, "Python programming language." in *USENIX annual technical conference*, vol. 41, 2007, p. 36.

[13] *Virtual Switch Acceleration with OVS-TC*, Jul 2018. [Online]. Available: https://www.netronome.com/m/documents/WP_OVS-TC_.pdf

[14] *Netronome Agilio CX Series*. [Online]. Available: https://www.netronome.com/products/agilio-cx/

[15] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar *et al.*, "The design and implementation of open vswitch," in *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, 2015, pp. 117–130.

[16] A. Voellmy, Y. R. Yang, and X. Shi, "Towards automatic generation of multi-table datapath from datapath-oblivious algorithmic sdn policies."