

3.2: z/OS Communications Server Performance Summary Report

Kaji Rashad

March 5, 2026

Contents

1	Trademarks, Notices, and Disclaimers	4
2	Acknowledgements	5
3	Tips for Reading This Document	6
4	Preface	7
5	Hardware Information	8
6	Workload Naming Convention	9
6.1	Introduction	9
6.2	Generic Workloads	9
6.3	Examples	9
6.4	CPU Cost/Tran And Transaction [Trans/sec]	9
7	Performance Best Practices: General	10
7.1	INBPERF DYNAMIC	10
7.2	QDIO Inbound Workload Queueing (WORKLOADQ)	10
7.3	IPCONFIG SEGMENTATIONOFFLoad (LSO)	10
7.4	TCPCONFIG AUTODELAYAck	10
7.5	IPCONFIG QDIOACCErator	10
7.6	MSG.WAITALL	10
8	Performance Best Practices: Security	12
8.1	HEAPPOOLS64	12
8.1.1	TLSv1.3 & TLSv1.2: With Versus Without HP64	12
9	Performance Best Practices: Workload(s) Reading From Storage Devices	13
9.1	High Performance FICON for IBM z Systems (zHPF)	13
10	Network Express on z17	14
10.1	New Network Express Feature!	14
10.1.1	Background	14
10.1.2	Testing Environment: CHPID Type OSH vs OSD	14
10.1.3	z/OS Environment Configuration: CHPID Type OSH vs OSD	14
10.1.4	Results	15
10.1.5	Observations	16
10.2	Sharing The New Network Express!	17
10.2.1	Background	17
10.2.2	Testing Environment: Shared OSH versus HiperSockets	17
10.2.3	z/OS Environment Configuration: Shared OSH, Shared OSD, and HiperSockets	17

10.2.4	Results	18
10.2.5	Observations	19
10.3	Enterprise Extender (EE) Over The New Network Express!	20
10.3.1	Background	20
10.3.2	Testing Environment: EE Over OSH versus OSD	20
10.3.3	z/OS Environment Configuration: EE Over OSH versus OSD	20
10.3.4	Results	20
10.3.5	Observations	21
10.4	GRE IWQ Performance on OSH!	21
10.4.1	Background	21
10.4.2	Testing Environment: GRE IWQ on OSH	21
10.4.3	z/OS Environment Configuration: GRE IWQ on OSH	21
10.4.4	Results	22
10.4.5	Observations	22
10.5	Network Express IWQ: New IP Router Queue!	23
10.5.1	Background	23
10.5.2	Testing Environment: IP Router	23
10.5.3	z/OS Environment Configuration: IP Router	23
10.5.4	Results	24
10.5.5	Observations	25
10.6	SMC-Rv2 Over NETH	26
10.6.1	Background	26
10.6.2	Testing Environment: OSD + RNIC vs NETH	26
10.6.3	z/OS Environment Configuration: OSD + RNIC vs NETH	26
10.6.4	Observations	26
10.7	Network Express: TCP/IP Traffic Over OSH & SMC-Rv2 Traffic Over NETH	27
10.7.1	Background	27
10.7.2	Testing Environment: TCP/IP Traffic Over OSH & SMC-Rv2 Traffic Over NETH	27
10.7.3	z/OS Environment Configuration: TCP/IP Traffic Over OSH & SMC-Rv2 Traffic Over NETH	27
10.7.4	Results	28
10.7.5	Observations	29
10.8	Shared Network Express between z/OS & Linux on Z	30
10.8.1	Background	30
10.8.2	Testing Environment: Shared Network Express between z/OS & Linux on Z	30
10.8.3	z/OS Environment Configuration: Shared Network Express between z/OS & Linux on Z	30
10.8.4	Results	31
10.8.5	Observations	32
10.9	Network Express: TCP/IP versus SMC-Rv2	33
10.9.1	Background	33
10.9.2	Testing Environment: TCP/IP versus SMC-Rv2	33
10.9.3	z/OS Environment Configuration: TCP/IP versus SMC-Rv2	33
10.9.4	Results	34
10.9.5	Observations	35
11	Hardware Performance: z17	36
11.1	HiperSockets: z17 vs z16	36
11.1.1	Background	36
11.1.2	Testing Environment: HiperSockets on z17	36
11.1.3	z/OS Environment Configuration: HiperSockets on z17	36
11.1.4	Results	37
11.1.5	Observations	37
11.2	SMC-Dv2: z17 vs z16	37

11.2.1	Background	37
11.2.2	Testing Environment: SMC-Dv2 on z17	37
11.2.3	z/OS Environment Configuration: SMC-Dv2 on z17	37
11.2.4	Results	38
11.2.5	Observations	39
11.3	z17: HiperSockets vs SMC-Dv2	39
11.3.1	Background	39
11.3.2	Testing Environment: HiperSockets vs SMC-Dv2	39
11.3.3	z/OS Environment Configuration: HiperSockets vs SMC-Dv2	39
11.3.4	Results	40
11.3.5	Observations	41
12	Network Security: AT-TLS	42
12.1	AT-TLS: Legacy RSA Signature Pair Against RSASSA-PSS Signature Pair	42
12.1.1	Background	42
12.1.2	Testing Environment: Legacy RSA Signature Pair Against RSASSA-PSS Signature Pair	42
12.1.3	z/OS Environment Configuration: Legacy RSA Signature Pair Against RSASSA-PSS Signature Pair	42
12.1.4	Results	43
12.1.5	Observations	43
12.2	AT-TLS: AES-CBC versus AES-GCM	43
12.2.1	Background	43
12.2.2	Testing Environment: AES-CBC versus AES-GCM	43
12.2.3	z/OS Environment Configuration: AES-CBC versus AES-GCM	44
12.2.4	Results	44
12.2.5	Observations	44
12.3	AT-TLS: TLSv1.3 vs TLSv1.2	45
12.3.1	Background	45
12.3.2	Testing Environment: TLSv1.3 vs TLSv1.2	45
12.3.3	z/OS Environment Configuration: TLSv1.3 vs TLSv1.2	45
12.3.4	Results	46
12.3.5	Observations	46
13	TLS Performance Paper	46
14	ICSF Optimization	46
14.0.1	Background	46
14.0.2	Testing Environment: TOKENOBJ(YES) & SESSIONOBJ(NO)	46
14.0.3	z/OS Environment Configuration: TOKENOBJ(YES) & SESSIONOBJ(NO)	47
14.0.4	Observations	48
15	3.2 versus 3.1: Release To Release Comparison	49
15.1	z17: z/OS Communications Server 3.2 versus 3.1	49
15.1.1	Introduction	49
15.1.2	Testing Environment: z/OS Communications Server 3.2 versus 3.1	49
15.1.3	z/OS Environment Configuration: z/OS Communications Server 3.2 versus 3.1	49
15.1.4	Observations	49
16	z/OS Communications Server Performance Index	49
	References	50

1 Trademarks, Notices, and Disclaimers

No part of this document may be reproduced or transmitted in any form without written permission from IBM Corporation. Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This information may include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or programs(s) at any time without notice. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED AS IS WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. IBM is not responsible for the performance or interoperability of any non-IBM products discussed herein.

The performance data contained herein was obtained in a controlled, isolated environment. Actual results that may be obtained in other operating environments may vary significantly. While IBM has reviewed each item for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Statements regarding IBM's future direction and intent are subject to change or withdraw without notice, and represent goals and objectives only. The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The most current copyright and trademark information are located here: <https://www.ibm.com/legal/copyright-trademark#trademarks>

Copyright © 2026 by International Business Machines Corporation

2 Acknowledgements

The z/OS Communications Server Performance Team would like to thank the following IBMers for their input on this report:

David Herr — Senior Software Engineer @ z/OS Communications Server

Christopher Nyamful — Software Engineer @ z/OS Communications Server

Jack Yapi — Software Engineer @ z/OS Communications Server

Michael Fitzpatrick — Senior Technical Staff Member @ z/OS Communications Server

3 Tips for Reading This Document

1. Clicking on any row in the Table of Contents will take the reader to that specific section or subsection of the document
2. All hyperlinks redirect to an external webpage or internal section/sub-section

4 Preface

The performance measurements discussed in this document were collected using dedicated system environments. Results obtained in other configurations or operating system environments may vary significantly depending upon environments used. Therefore, no assurance can be given, and there is no guarantee that an individual user will achieve performance or throughput improvements equivalent to the results stated here.

The Central Processor Unit (CPU) resources consumed listed includes only z/OS host networking related CPU costs (including dispatching costs) on **general Central Processors (CPs)** from the network device driver layer up through the application socket layer. The socket applications used in the micro-benchmarks for this publication have no application logic, so the CPU numbers represent the total application CPU cost which in this case equates to the network related CPU costs. With typical production workloads, network related cost compared to the overall application's transaction cost will vary based on the type of workload. Network cost is typically a small percentage of the overall CPU cost for transactional workloads but can have a larger cost for streaming workloads.

Note 1: In all benchmarks, the best practices recommended by z/OS Communications Server were utilized when applicable. The following best practices represent configuration options not enabled by default. Other options representing best practices that are enabled by default are not listed here (e.g., IPCONFIG CHECKSUMOFFLoad, READSTORAGE GLOBAL¹, GLOBALCONFIG ADJUSTD-Vipamss, etc.):

VTAM START Option:

1. QDIOSTG = 126

TCP/IP Configuration Profile:

1. INBPERF DYNAMIC WORKLOADQ (IWQ)

This is the default for Network Express CHPID Type OSH

Client and Server Side:

1. IPCONFIG SEGMENTATIONOFFLoad (LSO)
2. IPCONFIG6 SEGMENTATIONOFFLoad (LSO)
3. IPCONFIG QDIOACCEerator
4. TCPCONFIG AUTODELAYAck
5. MSG_WAITALL ²

The reader should read [1] for a description of the z/OS Communications Server and **OSA-Express** best practices. For convenience, you can access it via this link: <https://www.ibm.com/support/pages/node/6562173>.

¹READSTORAGE GLOBAL is an option on the INTERFACE statement

²A socket read flag utilized by the application to instruct the TCP layer to delay completion of a Socket Receive or Read call until the full length of the requested data is available in the TCP receive buffer

5 Hardware Information

z17

Machine Type (Model): 9175 - ME1

z16

Machine Type (Model): 3931 - A01

6 Workload Naming Convention

6.1 Introduction

Readers can decipher the listed workloads in the following way: [NameOfBenchmark][NumberOfClients] (BytesSentByClient/BytesSentByServer). For example, RR(10)(1B/100B) is interpreted as Request Response benchmark with 10 clients sending 1 byte and receiving 100 bytes from the server.

6.2 Generic Workloads

RRx(y/z): x number of clients doing Request Response transactions where the client is opening a connection and performing a series of transactions sending y bytes and receiving a response of z bytes

RRx(y): x number of clients doing Request Response transactions where the client is opening a connection and performing a series of transactions sending and receiving a response of y bytes

CRRx(y/z): x number of clients doing Connect Request Response transactions where the client is performing a series of repeatable transactions consisting of opening a connection, sending y bytes, receiving a response of z bytes, and closing the connection

STRx(y/z): x number of clients doing Streaming transactions where the client is opening a connection and performing a series of transactions sending y bytes and receiving a response of z bytes

6.3 Examples

RR40(100B): In an instance of time, there are 40 clients that establish secure shell (SSH) connections to a server then performs request of 100 bytes before getting a response of 100 bytes. The request & response pattern continues with the connections being in established state as long as the SSH session is active. The connections are terminated when the SSH sessions are terminated.

CRR9(64B/200B): In an instance of time, there are 9 clients sending a HTTP GET request containing 64 bytes and receives a response containing 200 bytes which allows them to log into their bank portal. The core difference between a RR and CRR workload is the duration of the connection. In CRR, the connection is closed after each transaction. A common use case for a bank portal is logging in to audit the balance before logging out.

STR3(1B/20MB): In an instance of time, there are 3 clients sending a 1 byte request and receiving a 20MB file in response simulating what is done with typical bulk-transfer services such as File Transfer Protocol (FTP).

6.4 CPU Cost/Tran And Transaction [Trans/sec]

On some graphs, the reader will observe a key legend of “CPU Cost/Tran” and “Transaction [Trans/sec]”. Our measurement uses the z/OS Resource Measurement Facility (RMF) to determine the average CPU utilization [2]. RMF results show the CPU utilization across all online CPs during a sampling interval which is taken into consideration when performing our calculations.

For example, if RMF result shows a LPAR utilization of 25% across 4 CPs then we translate this into 100% of 1 CP. If the sampling interval is 10 seconds and we are averaging 100% of 1 CP then the benchmark consumes 10 seconds of CPU during the sampling period. If there were 1 million transactions during the 10 second sampling interval, then there was a transaction rate of 1,000,000 trans / 10 seconds (or 100,000 trans/sec) and a CPU cost per transaction of (10 CPU seconds) / (1,000,000 transactions) or 10 [μs]/tran.

7 Performance Best Practices: General

7.1 INBPERF DYNAMIC

Processing inbound traffic for the OSA-Express interface in Queued Direct Input Output (QDIO) mode dynamically exploits an OSA hardware function called Dynamic LAN Idle. The DYNAMIC setting reacts to changes in traffic patterns and dynamically sets the interrupt-timing values to optimize response times. Refer to [3] for more information.

Note: This setting only applies to OSA-Express because Network Express always uses DYNAMIC mode.

7.2 QDIO Inbound Workload Queueing (WORKLOADQ)

The core benefits of Inbound Workload Queueing (IWQ) are “finer tuning of read-side interrupt frequency to match the latency demands of the various workloads that are serviced” and “improved multiprocessor scalability as multiple OSA-Express input queues are efficiently serviced in parallel” [4]. Each queue is tailored for its specific need. For instance, the bulk queue is tailored for improved “in-order packet delivery on multiprocessor, which likely results in improvements to CPU consumption and throughput” [4]. QDIO IWQ provides benefit on both sides of the connection hence it is enabled on both sides in our test set-up when applicable. Note that WORKLOADQ requires the processing of inbound traffic for the QDIO interface to be set as DYNAMIC (e.g., INBPERF DYNAMIC WORKLOADQ). Refer to [4] for more information.

Note: This setting only applies to OSA-Express because Network Express always uses WORKLOADQ (i.e., IWQ).

7.3 IPCONFIG SEGMENTATIONOFFLoad (LSO)

Any large amount of data traveling over the network is broken down into smaller segments by the TCP/IP stack. This process can be CPU intensive. As an alternative, segmentation offload (i.e., Large Send Offload) is an OSA-Express and Network Express feature. It reduces host CPU utilization and increases data transfer efficiency by offloading segmentation processing to OSA [5].

7.4 TCPCONFIG AUTODELAYAck

Reduction in network traffic and CPU utilization can be achieved by delaying the TCP acknowledgement (ACK) depending on the traffic pattern. AUTODELAYAck enables the TCP/IP stack to “automatically enable or disable a delayed ACK in a TCP connection based on the characteristic of the traffic” [6].

7.5 IPCONFIG QDIOACCEerator

QDIO Accelerator specifies that inbound packets that are to be forwarded by a TCP/IP stack are eligible to be routed directly between any of the following combinations of interface types: a HiperSockets interface and an OSA-Express QDIO interface, two OSA-Express QDIO interfaces, and two HiperSockets interfaces. These packets arrive at the forwarding stack, but do not traverse all the TCP/IP layers for forwarding. Therefore, system resources (storage and CPU) are not expended for purposes of routing and forwarding packets. This option also applies to packets that would be forwarded by the Sysplex Distributor. Refer to [7] more information.

Note: QDIOAccelerator for Network Express supports acceleration of traffic from and to EQDIO interfaces only. For EQDIO interfaces, no other combinations are supported.

7.6 MSG_WAITALL

MSG_WAITALL is beneficial in streaming workloads. The flag bit decreases the frequency of signaling occurring for the application receiving data as less signals can result in improvements to CPU consumption and throughput. The receiving application is signaled only when all requested data can

be returned. To avoid blocking the application indefinitely, the flag bit should only be set in scenarios where the application expects to receive enough data to fill its buffer, or the connection will terminate.

8 Performance Best Practices: Security

8.1 HEAPPOOLS64

8.1.1 TLSv1.3 & TLSv1.2: With Versus Without HP64

Application Transparent Transport Layer Security (AT-TLS) creates System SSL environments using the z/OS Language Environment (LE). These System SSL environments use the LE runtime default options or those specified in the CEEPRMxx PARMLIB member. The default LE runtime does not have HEAPPOOLS64 enabled. For large AT-TLS configurations, running without HEAPPOOLS64 enabled could result in additional contention for user heap storage across the different System SSL environments. This could lead to slow-downs or timeouts processing TLS handshakes. The LE heap contention issue is also a big factor when you use TLSv1.3 regardless of the workload. By enabling the HEAPPOOLS64³ runtime option, this contention for user heap storage can be eliminated. The z/OS V2R5 Communications Server Performance Summary report [9] shows HEAPPOOLS64 has a positive impact on connections using AT-TLS. In the report section, *Performance Best Practice: Security*, the positive impact was measured across all TLSv1.2 and TLSv1.3 ciphers. z/OS Communications Server APAR PH59425 now ensures HEAPPOOLS64 is always enabled. If you are on z/OS V2R5 or z/OS 3.1 then ensure z/OS Communications Server APAR PH59425 is applied. From z/OS 3.2 and onward, HEAPPOOLS64(ON) became the default option.

³Enabling HEAPPOOLS64 is referring to coding HEAPPOOLS64(ON) in the TCP/IP procedure where the default values were taken for each cell size and its associated pool count [8].

9 Performance Best Practices: Workload(s) Reading From Storage Devices

9.1 High Performance FICON for IBM z Systems (zHPF)

High Performance FICON for IBM z Systems (zHPF) utilizes the Fibre Channel Protocol (FCP) which is a high-speed data transfer mechanism between many different objects such as workstations, mainframes, supercomputers, storage devices, and displays [10]. For our purpose, the objects of interest are mainframes and storage devices. *zHPF increases the number of I/O per second between mainframes and storage devices.* In the z/OS 3.1 Communications Server Performance Summary report [11], section *Performance Best Practice: Workload(s) Reading From Storage Devices*, the benefit of using zHPF is exemplified.

10 Network Express on z17

10.1 New Network Express Feature!

10.1.1 Background

The latest Central Processor Complex (CPC) model, z17, supports a new Network Express feature. The new Network Express feature provides both the latest Enhanced QDIO (EQDIO) architecture for reliable high-speed Ethernet transport and RoCEv2 support for optimized TCP connectivity using RDMA technology with Shared Memory Communications (SMC-R). Network Express supports two different modes:

1. CHPID Type OSH: Supports the EQDIO architecture allowing for reliable high-speed Ethernet transport
2. FID Type NETH: Supports Remote Direct Memory Access (RDMA), RDMA Over Converged Ethernet (RoCE), and SMC-R allowing for RoCEv2 support for optimized TCP connectivity using RDMA technology with Shared Memory Communications (SMC-R)

The reader is encouraged to read [12] [13] [14] to gather more details about the new Network Express feature.

10.1.2 Testing Environment: CHPID Type OSH vs OSD

The goal of this test was to benchmark a request response workload running over the existing OSA-Express feature adapter then compare against the new Network Express feature adapter running in OSH mode.

10.1.3 z/OS Environment Configuration: CHPID Type OSH vs OSD

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- Interfaces
 - Network Express CHPID Type OSH 25GbE
 - OSA-Express 7S CHPID Type OSD 25GbE
- MTU: 1500 (Regular Frames)

10.1.4 Results

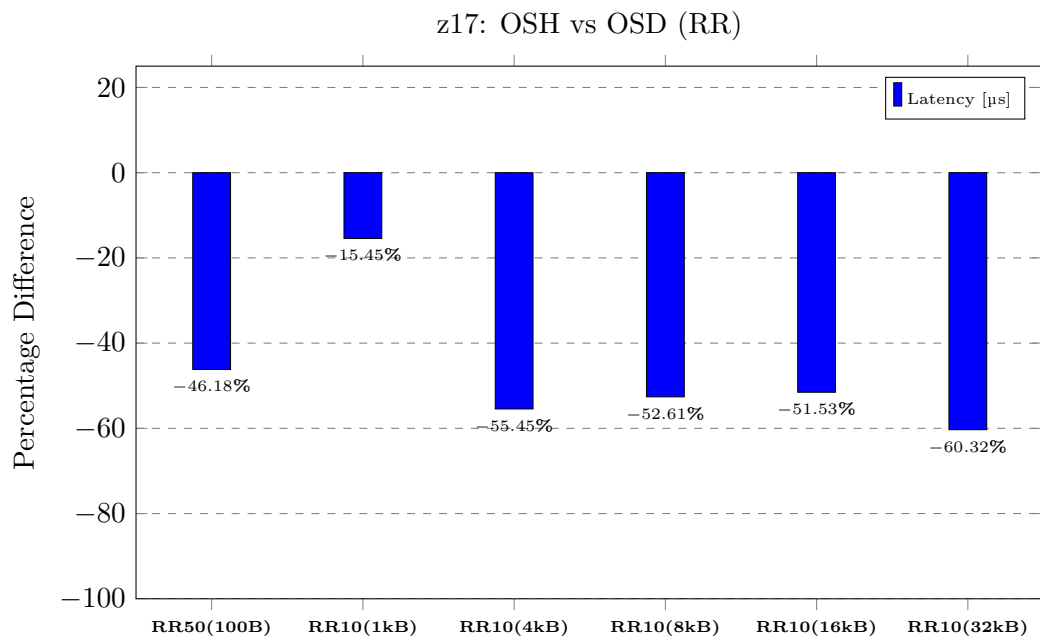


Figure 1: This graph represents running different interactive workloads over OSH and OSD then comparing the differences in network response times (network latency). The graph is answering the following question: *How much faster is OSH for interactive workload?*

z17: OSH vs OSD (STR)

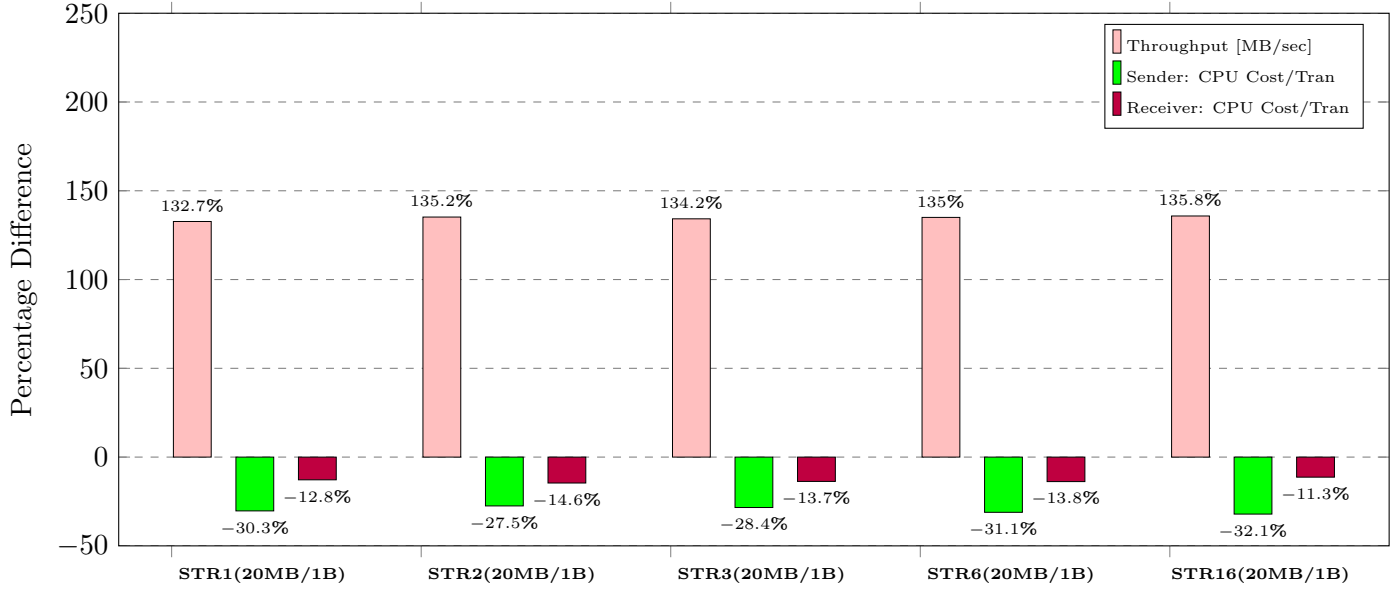


Figure 2: This graph represents running different streaming workloads over OSH and OSD then comparing the differences in network response times (network latency) and network CPU cost. The graph is answering the following question: *How much faster and more efficient in using system resources (CPU) is OSH for streaming workload?*

10.1.5 Observations

In reference to fig. 1 and fig. 2, the following conclusions can be drawn:

- A request response workload going over Network Express instead of OSA-Express on z17 will decrease the transaction workload network latency
- A streaming workload going over Network Express instead of OSA-Express on z17 will:
 - Increase the throughput rate
 - Decrease the sender network CPU cost
 - Decrease the receiver network CPU cost

10.2 Sharing The New Network Express!

10.2.1 Background

It is possible to share the Network Express feature just like you can share the OSA-Express feature. There are two types of sharing. *Internal sharing* consist of two intra-CPC LPARs using the same network interface controller (NIC) where the communication traffic **does not** get routed through the switch nor router. In comparison, *external sharing* consist of two intra-CPC LPARs using the same NIC where the communication traffic **does** route through the switch and/or router. In reference to [15], OSA-Express had *internal sharing* limitations when processing streaming workload patterns between any two intra-CPC LPARs. The limitations were eradicated with the new Network Express feature.

10.2.2 Testing Environment: Shared OSH versus HiperSockets

The goal of this test was to benchmark a streaming workload running over an internally shared Network Express feature then comparing it against HiperSockets to exemplify how the new Network Express feature is ready to handle any streaming workload occurring between any two intra-CPC LPARs. Also, the testing will showcase how OSH in shared mode is capable of matching HiperSockets in terms of network latency and network CPU cost. The following figure shows the test environment set-up.

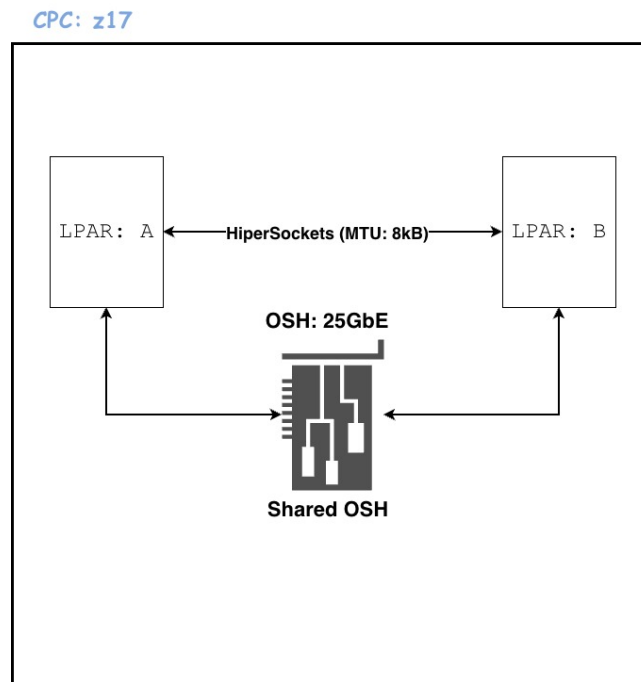


Figure 3: Test set-up for gathering internal shared OSH versus HiperSockets metrics

10.2.3 z/OS Environment Configuration: Shared OSH, Shared OSD, and HiperSockets

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- Interfaces
 - Network Express CHPID Type OSH 25GbE
 - HiperSockets

- MTU: 9000 (Jumbo Frames)
- HiperSockets Maximum Frame Size: 16KB
TCP/IP MTU Size: 8KB

10.2.4 Results

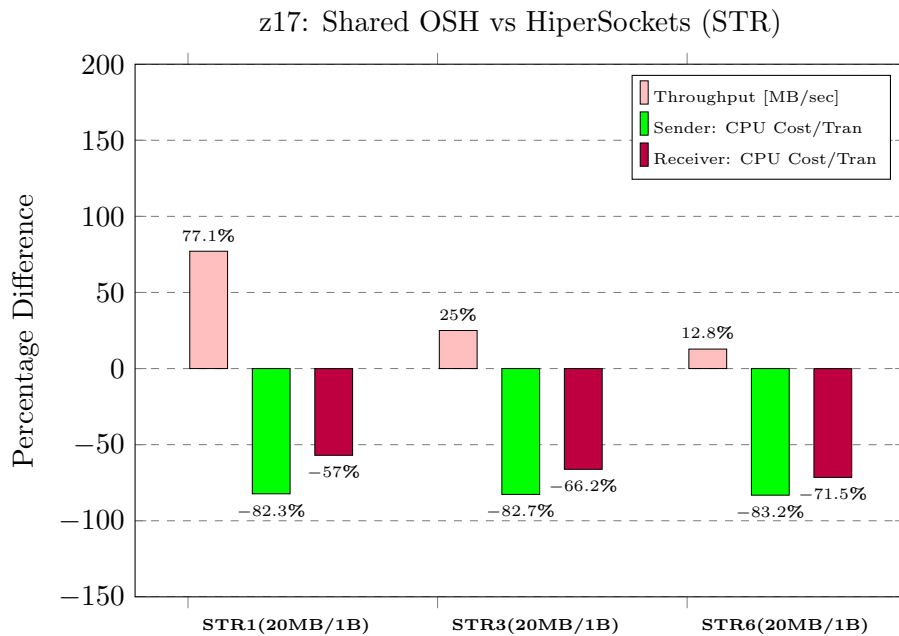


Figure 4: This graph represents the comparison of two intra-CPC LPARs streaming data over an internal shared OSH versus HiperSockets. This graph answers the following question: *How much faster and cheaper is internal shared OSH compared to HiperSockets for streaming data between two intra-CPC LPARs?*

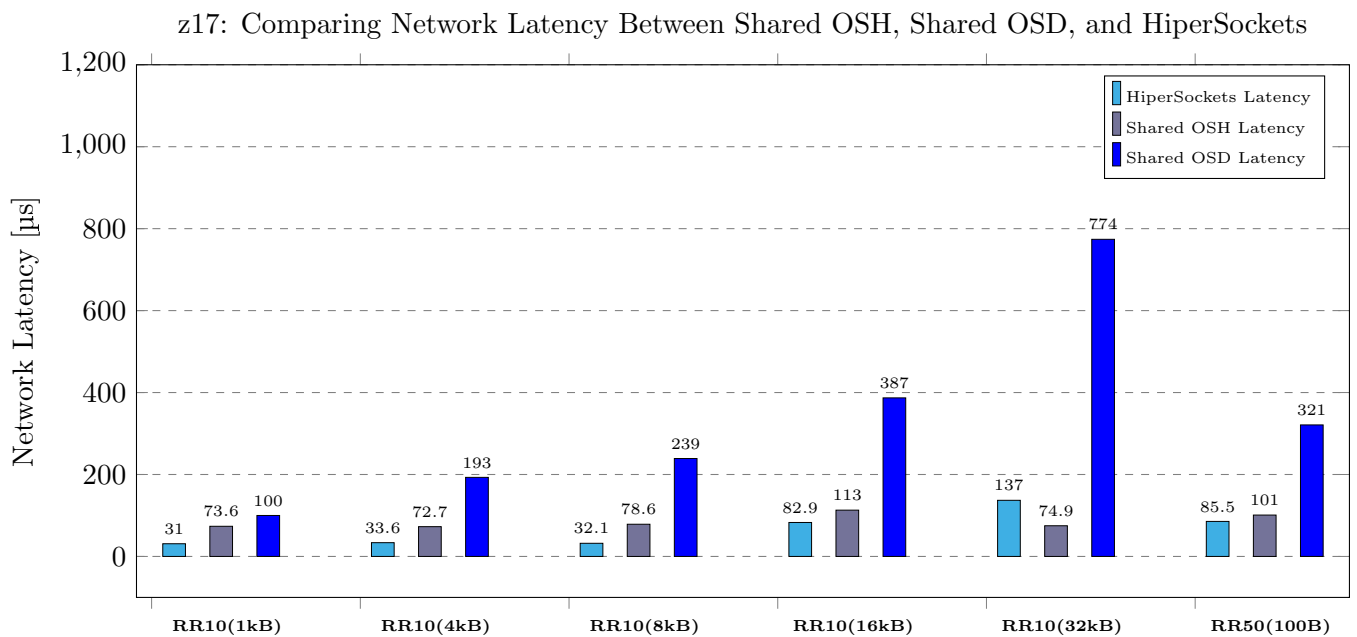


Figure 5: This graph compares the network latency between Shared OSH, Shared OSD, and HiperSockets for multiple different interactive workloads

10.2.5 Observations

In reference to fig. 4 and fig. 5, the following conclusions can be drawn:

- Running a streaming workload over a shared OSH is now possible where it had performance limitation over shared OSD. Not only is it possible, it offers significant network CPU cost savings with lower network latency as shown in fig. 4 when compared to HiperSockets.
- Running interactive workloads over shared OSH, shared OSD, and HiperSockets revealed the following network latency observations:

Shared OSH becomes comparable with HiperSockets as the payload increases then shows an improvement over HiperSockets network latency when the payload becomes 32kB

Shared OSH always does better than Shared OSD starting from small to large payloads. Shared OSH performs significantly better with large payloads as evident by a payload size of 32kB.

In general, shared OSH showed an average of 30% network CPU cost reduction for interactive work when compared against HiperSockets

10.3 Enterprise Extender (EE) Over The New Network Express!

10.3.1 Background

Enterprise Extender (EE) is a standard that is documented in RFC 2353. It extends SNA High Performance Routing (HPR) traffic of any logical unit over an IP infrastructure transparently without any required infrastructure changes. Such offering makes EE attractive to end users (i.e., clients). For more information, refer to [16].

10.3.2 Testing Environment: EE Over OSH versus OSD

The goal of this test was to benchmark interactive workloads using EE over Network Express feature and OSA-Express feature then make a comparison.

10.3.3 z/OS Environment Configuration: EE Over OSH versus OSD

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- Interfaces
 - Network Express CHPID Type OSH 25GbE
 - OSA-Express 7S CHPID Type OSD 25GbE

10.3.4 Results

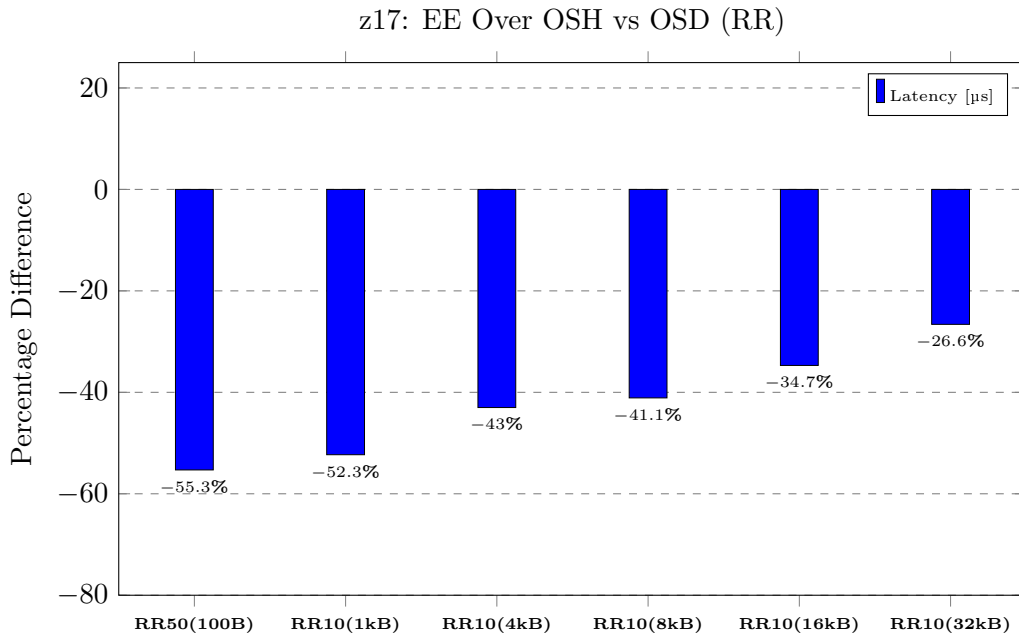


Figure 6: This graphs shows a comparison of running different types of interactive workload over EE on OSH & OSD then make a comparison. This graph answers the following question: *How much faster are my EE workloads over OSH?*

10.3.5 Observations

In reference to fig. 6, the following conclusion can be drawn:

- Running EE workloads over Network Express feature provides significant network latency reduction when compared to the same workload running over OSA-Express feature

10.4 GRE IWQ Performance on OSH!

10.4.1 Background

When a `VIPADYNAMIC VIPAROUTE` statement is configured, it indicates that Sysplex Distributor uses the IP routing tables to select the route to the target TCP/IP stack, rather than the default dynamic XCF interfaces. To successfully route the Sysplex Distributor traffic to a target, a Generic Routing Encapsulation (GRE) header is prepended to each IP packet. When these IP packets arrive at the target TCP/IP stack over OSA-Express feature (OSD), Inbound Workload Queueing (IWQ) rules are not applied, so all IP packets are delivered to the primary queue. Any optimizations by separating bulk inbound traffic from interactive inbound traffic does not take place. With Network Express feature (OSH), IWQ rules can now be applied, even if these IP packets arrive prepended with a GRE header. This enables Sysplex Distributor traffic for mixed workloads (e.g., RR + STR) to benefit from being delivered onto different inbound queues.

10.4.2 Testing Environment: GRE IWQ on OSH

The goal of this test was to benchmark the network CPU cost reduction offered by the new GRE IWQ feature offered solely on Network Express CHPID Type OSH. Here are some important notes regarding our testing:

- QDIOAccelerator was disabled on the distributing stack for fair comparison
It created a fair comparison as both tests used in the comparison did not use QDIOAccelerator. Without disabling QDIOAccelerator, the test where GRE IWQ was not being used would have utilized QDIOAccelerator since QDIOAccelerator is supported when the distributor and target are using OSD. However, the test using GRE IWQ would not be using QDIOAccelerator as the distributor was using OSD but the target had OSH. QDIOAccelerator is not supported when the sender and receiver are of different NIC types.
- Mixed workloads (e.g., RR + STR) ran at the same time through the Sysplex Distributor node and routed to the target stack

10.4.3 z/OS Environment Configuration: GRE IWQ on OSH

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- Workloads
RR10(4kB)
STR1(5MB/1B)
- Interfaces
Network Express CHPID Type OSH 25GbE
OSA-Express 7S CHPID Type OSD 25GbE

10.4.4 Results

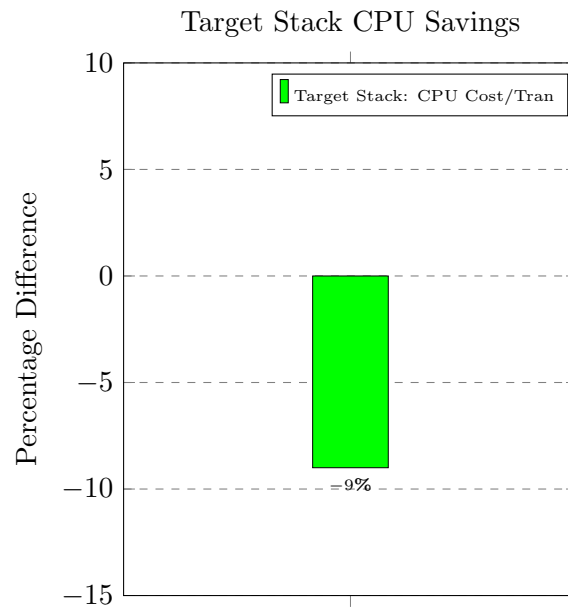


Figure 7: This graph shows a comparison of running mixed workload through Sysplex Distributor where the client, distributor and target used CHPID Type OSD then compares to a test where the target used CHPID Type OSH. This graph answers the following question: *If I had the target stack use CHPID Type OSH instead of OSD then how much network CPU cost savings will I get from the GRE IWQ feature which separates incoming GRE encapsulated streaming packets into the bulk queue?*

10.4.5 Observations

In reference to fig. 7, the following conclusion can be drawn:

- Network Express CHPID Type OSH uses GRE IWQ to separate streaming traffic into the bulk queue at the target stack. Such separation reduces the CPU cost at the target stack by nine percent as the GRE IWQ reduces packets arriving out of order for the streaming workload.

10.5 Network Express IWQ: New IP Router Queue!

10.5.1 Background

The new Network Express feature has a new ancillary input queue (AIQ) called IP Router. This new design allows for IP forwarded traffic separation from the primary queue. Such separation allows QDIOAccelerator processing to be targeted at the IP Router and Sysplex Distributor AIQs hence avoiding the cost of accelerator processing on the primary queue. Refer to [17] for more information.

10.5.2 Testing Environment: IP Router

The goal of this test was to benchmark the network CPU cost reduction and network latency decrease offered by the new IP Router AIQ. Here are some important notes regarding our testing:

- The IP Router input queue is created for a Network Express feature when:
 - Datagram forwarding is enabled (DATAGRAMFWD on the IPCONFIG statement)
 - QDIOAccelerator is enabled (QDIOACCELERATOR on the IPCONFIG statement),
 - ROUTEALL is configured for this interface (VMAC ROUTEALL (default option) on the INTERFACE statement)

10.5.3 z/OS Environment Configuration: IP Router

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- Interfaces
 - Network Express CHPID Type OSH 25GbE
 - OSA-Express 7S CHPID Type OSD 25GbE

10.5.4 Results

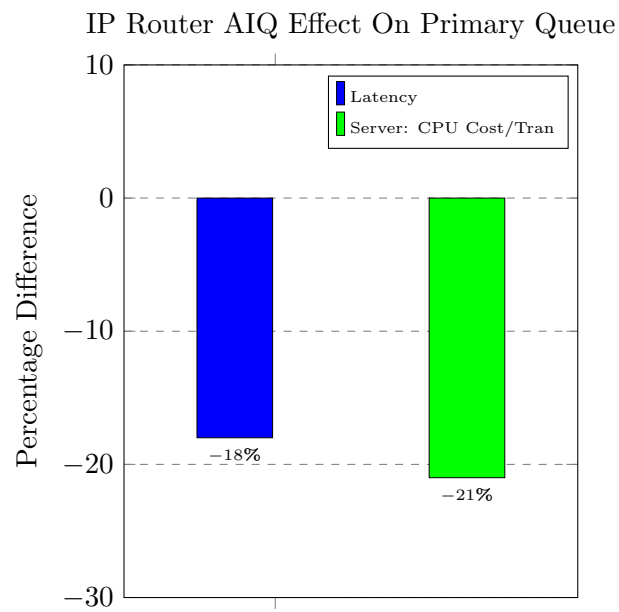


Figure 8: This graph compares two scenarios: IP Router activated versus IP Router deactivated. This graph shows how having the IP Router activated benefits the primary queue as it does not have to perform a look-up to understand whether the incoming packet is destined for itself or not. The graph answers the following question: *How much network latency and CPU cost is reduced when the IP router is activated?*

IP Router AIQ Effect On Local & Forwarded Traffic

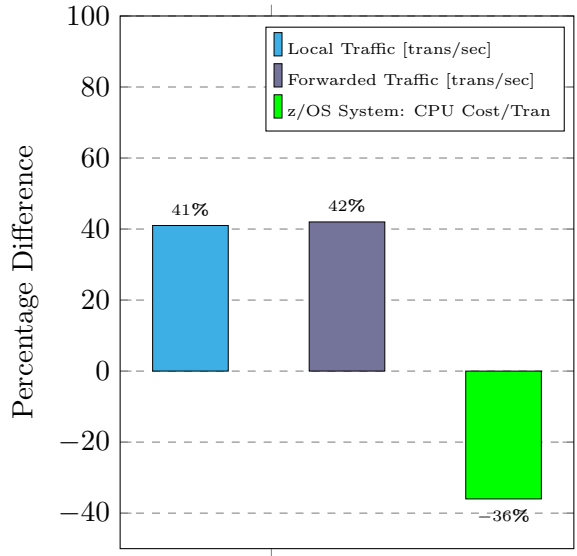


Figure 9: This graph is showing the comparison of using the IP Router AIQ to forward traffic instead of the primary queue. This graph answers the following questions: *If I use the IP Router AIQ to forward traffic then how many more incoming local transactions and forwarding transactions can I handle? Also, how much networking CPU cost does the z/OS system save by using the IP Router AIQ?*

10.5.5 Observations

In reference to fig. 8 and fig. 9, the following conclusions can be drawn:

- fig. 8 shows how a z/OS system processing inbound packets destined to itself will see benefit from the IP Router AIQ

The workload will see network latency and CPU cost reduction because it does not have to perform a look-up in deducing whether the inbound packet should be forwarded or not

- fig. 9 shows how an z/OS system processing a inbound mixed workload where some of the inbound packets are destined for itself versus some are forwarded to other z/OS systems benefits from the IP Router AIQ

Inbound packets that are destined for this z/OS system see an increase in its network transaction rate

The processing of network transactions that are being forwarded by the z/OS system increases

The z/OS system that is processing the mixed workload sees a decrease in its networking CPU cost

10.6 SMC-Rv2 Over NETH

10.6.1 Background

The SMC-Rv2 protocol is capable of using the new Network Express feature when it is configured in FID Type NETH. In reference to [12], it is possible to have a single Network Express NIC where each port is configured in two different modes:

1. CHPID Type OSH
2. Function ID (FID) Type NETH

Such capability reduces infrastructure and maintenance cost while allowing access to the SMC-Rv2 protocol **without additional hardware purchase**.

10.6.2 Testing Environment: OSD + RNIC vs NETH

The goal of this test was to compare running the SMC-Rv2 protocol on a z16 CPC configured with OSA-Express 7S 25GbE paired with a RoCE Express3 25GbE running z/OS 3.2 against running the same protocol on z17 over the Network Express FID NETH.

10.6.3 z/OS Environment Configuration: OSD + RNIC vs NETH

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17 & z16
 - Machine Type (Model): 9175 - ME1
 - Machine Type (Model): 3931 - A01
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- Interfaces
 - OSA-Express 7S CHPID Type OSD 25GbE
 - RoCE Express3 25GbE
 - Network Express FID Type NETH 25GbE

10.6.4 Observations

The testing revealed the following observations:

- The SMC-Rv2 protocol using NETH on z17 versus OSD + RNIC on z16 averages a 10% network latency reduction for interactive workloads.
- The SMC-Rv2 protocol using NETH on z17 versus OSD + RNIC on z16 averages a 10% network CPU cost reduction for streaming workloads.

10.7 Network Express: TCP/IP Traffic Over OSH & SMC-Rv2 Traffic Over NETH

10.7.1 Background

As described in section 10.6, it is now possible to have workloads running over two different supported protocols (e.g., TCP/IP, SMC-Rv2) concurrently over two separate ports but on the same Network Express feature (i.e., NIC). This is a benefit due to the stated reasons before (e.g., less infrastructure cost, less maintenance cost, no additional hardware required to exploit SMC-Rv2).

10.7.2 Testing Environment: TCP/IP Traffic Over OSH & SMC-Rv2 Traffic Over NETH

The goal of this test was to benchmark the scenario where you configure each port on the new Network Express in two different modes: CHPID OSH & FID NETH. In the first test, interactive traffic used CHPID Type OSH whereas streaming traffic used FID Type NETH. In this test, the comparison consisted of running the two different workloads on two different Network Express feature then merging them onto a single Network Feature by using both ports. In the second test, the testing environment remained the same except the workloads changed. The workload became streaming over both CHPID Type OSH and FID Type NETH. In other words, the comparison consisted of running streaming work over two different protocols on two different Network Express features then merging them onto a single Network Feature by using both ports.

10.7.3 z/OS Environment Configuration: TCP/IP Traffic Over OSH & SMC-Rv2 Traffic Over NETH

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- Interfaces
 - Network Express CHPID Type OSH 25GbE
 - Network Express FID Type NETH 25GbE

Note: In test one, the above two network interfaces represent two different Network Express features. In test two, the above two interfaces represent two different ports on the same Network Express feature.

10.7.4 Results

RR & CRR Over TCP/IP and STR Over SMC-Rv2

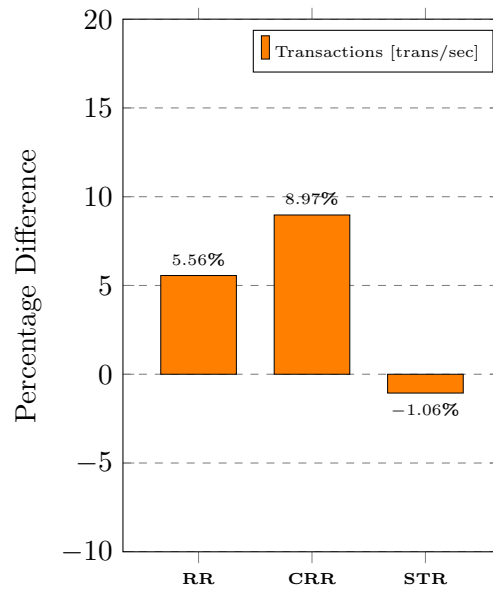


Figure 10: This graph shows the comparison of running interactive work over the TCP/IP protocol and streaming work over the SMC-Rv2 protocol where two dedicated Network Express feature were used in one instance versus another instance where a single Network Express feature was used. This graph answers the following question: *If I consolidate my interactive and streaming workload from using two dedicated NICs into a single NIC using both ports then is my network transaction rate impacted?*

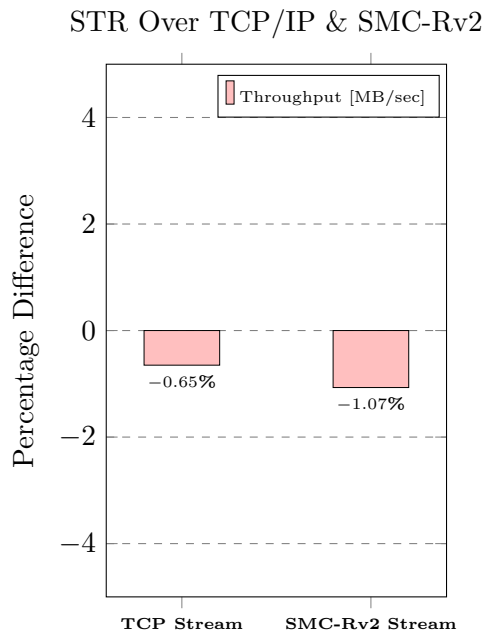


Figure 11: This graph shows the comparison of running streaming work over the TCP/IP protocol and SMC-Rv2 protocol where two dedicated Network Express feature were used in one instance versus another instance where a single Network Express feature was used. This graph answers the following question: *If I consolidate my streaming workload from using two dedicated NICs into a single NIC using both ports then is my network throughput rate impacted?*

10.7.5 Observations

fig. 10 and fig. 11 shows the following:

- Running interactive work over TCP/IP and streaming work over SMC-Rv2 where both protocols run on different ports on the same Network Express feature versus different Network Express feature showed no performance degradation
- Running streaming work over TCP/IP and SMC-Rv2 where both protocols run on different ports on the same Network Express feature versus different Network Express feature has minimal performance differences

10.8 Shared Network Express between z/OS & Linux on Z

10.8.1 Background

A Network Express feature can be configured as CHPID Type OSH and FID Typs NETH on z/OS at the same time. Linux uses FID Type NETH for both IP and SMC traffic. A throughput and networking CPU cost reduction benefit is observed for appliances using Linux on Z, such as IDAA, co-located with applications running on z/OS, such Db2, where the Network Express feature is internally shared between Linux on Z and z/OS. Refer to 10.2 for the definition of internal sharing.

10.8.2 Testing Environment: Shared Network Express between z/OS & Linux on Z

The testing goal was to make a comparison of having a zLinux client and z/OS server access to their own dedicated Network Express feature versus sharing the same port on a single Network Express feature between the LPARs. The following figure shows the testing environment where a zLinux client running RHEL 10 communicates with a z/OS server running z/OS 3.2. It also shows how each LPAR is connected to the same port but defined in different ways (e.g., OSH and NETH).

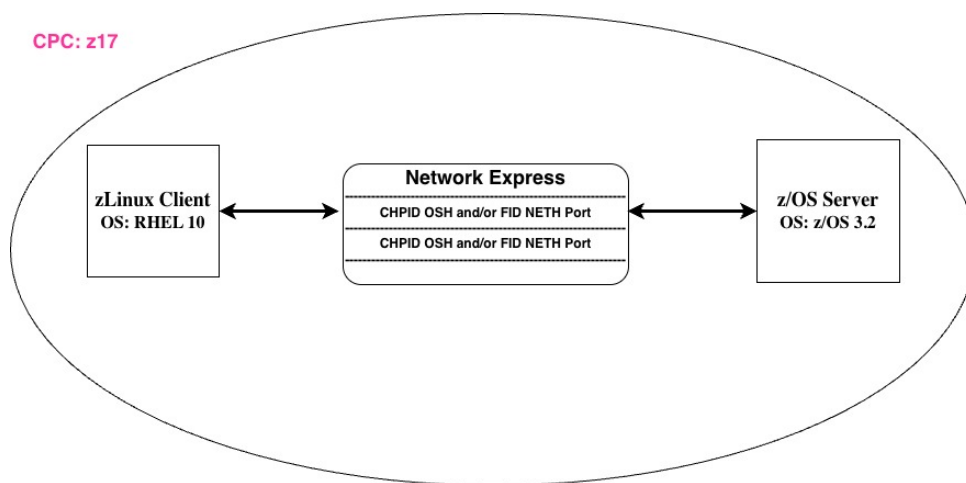


Figure 12: Test set-up where a zLinux client communicates with a z/OS server sharing the same port on the new Network Express feature

10.8.3 z/OS Environment Configuration: Shared Network Express between z/OS & Linux on Z

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 2 (Dedicated) per LPAR
- Interfaces

Network Express CHPID Type OSH 25GbE

Network Express FID Type NETH 25GbE

Note: In test one, the above two items represent two different Network Express features. In test two, the above two items represent the same port on the same Network Express feature.

10.8.4 Results

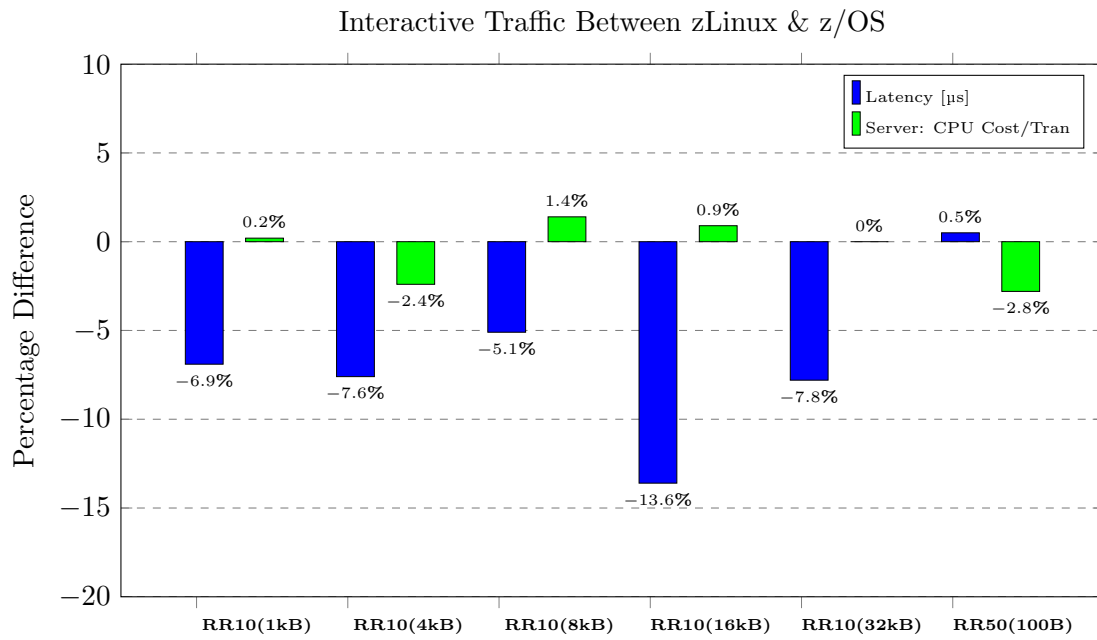


Figure 13: This graph shows the comparison of having a zLinux client and a z/OS server connected to their own dedicated Network Express feature then merging both client & server onto a single Network Express where they share a single port. This graph answers the following question: *If I have a zLinux application communicate with a z/OS server application where both use the same Network Express feature and communicate via internal sharing then what is the network latency reduction for interactive workloads?*

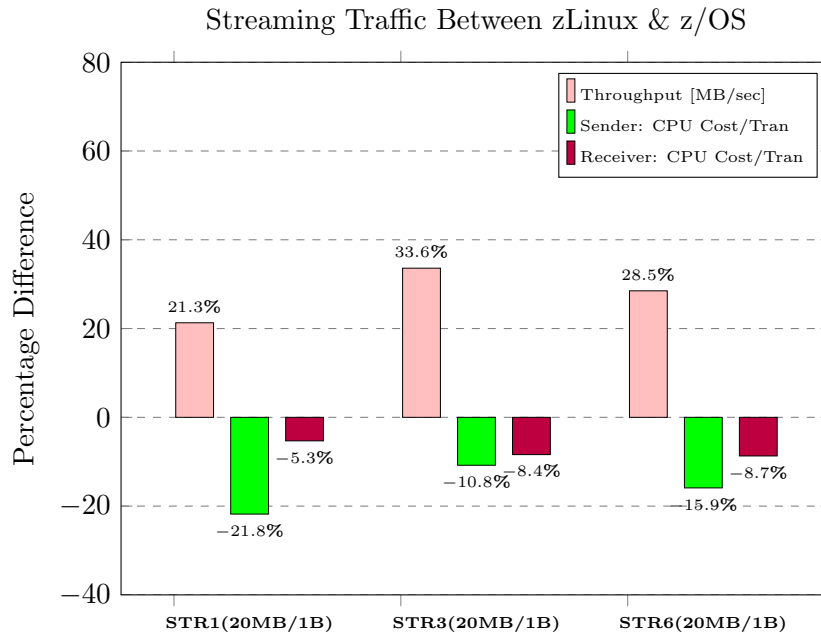


Figure 14: This graph shows the comparison of having a zLinux client and a z/OS server connected to their own dedicated Network Express feature then merging both client & server onto a single Network Express where they share a single port. This graph answers the following question: *If I have a zLinux application communicate with a z/OS server application where both use the same Network Express feature and communicate via internal sharing then what is the network throughput increase and network CPU cost savings for streaming workloads?*

10.8.5 Observations

fig. 13 and fig. 14 shows the following:

- A zLinux client application can communicate with a z/OS server application using the same Network Express port via internal sharing where:

Interactive workloads occurring between the two systems benefit from reduced networking latency

Streaming workloads occurring between the two systems benefit from higher throughput and less networking CPU cost for sender & receiver

10.9 Network Express: TCP/IP versus SMC-Rv2

10.9.1 Background

A Network Express feature can be configured as CHPID Type OSH and FID Type NETH on z/OS as described in section 10.1. This allows different businesses to exploit the SMC-Rv2 protocol **without any additional cost**.

10.9.2 Testing Environment: TCP/IP versus SMC-Rv2

The testing goal was to compare the TCP/IP protocol against the SMC-Rv2 protocol (i.e., RoCEv2 standard).

10.9.3 z/OS Environment Configuration: TCP/IP versus SMC-Rv2

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- Interfaces
 - Network Express CHPID Type OSH 25GbE
 - Network Express FID Type NETH 25GbE
- Workloads
 - RR60(4kB)
 - RR60(8kB)
 - RR60(16kB)
 - RR60(32kB)
 - STR1(1B/20MB)
 - STR1(20MB/1B)

10.9.4 Results

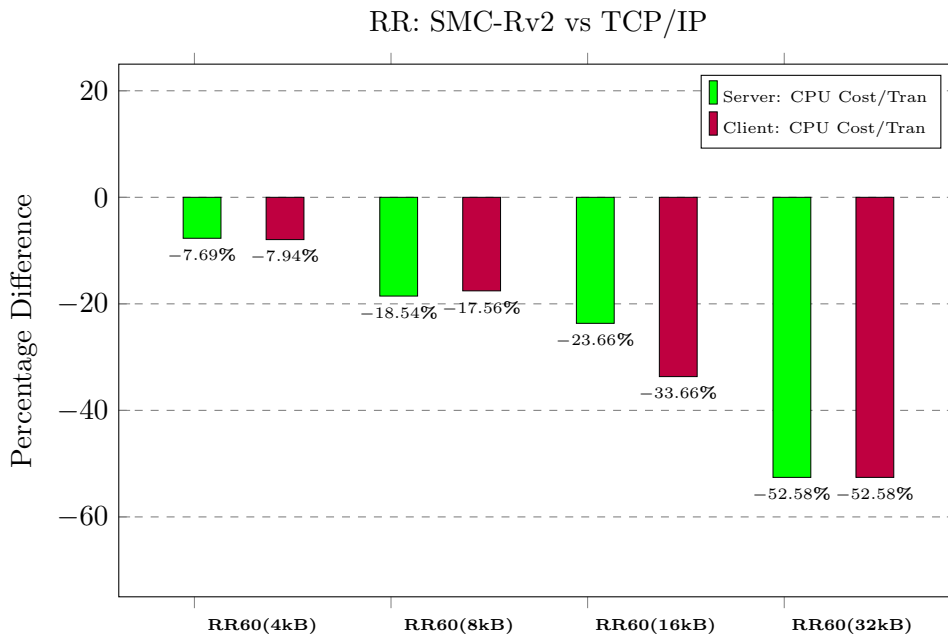


Figure 15: This graph shows the comparison of network CPU cost savings when running different interactive workloads over the SMC-Rv2 protocol versus the TCP/IP protocol. In other words, this graph answers the following question: *How much more efficient is the SMC-Rv2 protocol in using system resources (CPU) compared to TCP/IP for interactive work?*

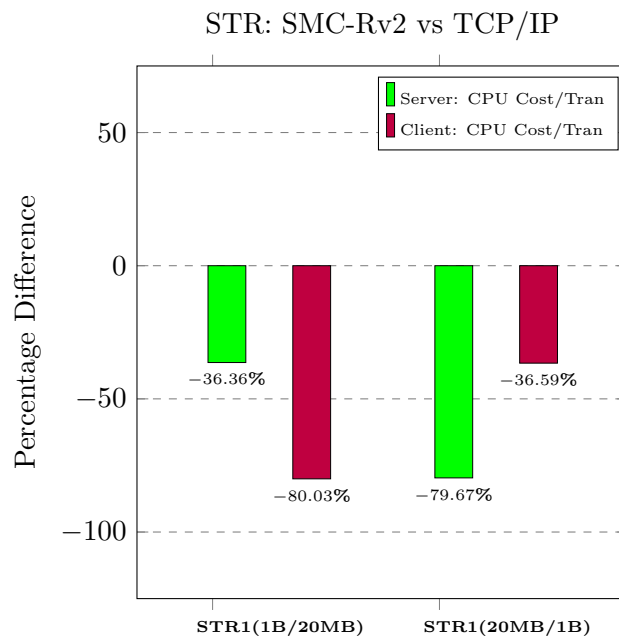


Figure 16: This graph shows the comparison of network CPU cost savings when running bulk transfer workloads over the SMC-Rv2 protocol versus the TCP/IP protocol. In other words, this graph answers the following question: *How much more efficient is the SMC-Rv2 protocol in using system resources (CPU) compared to TCP/IP for streaming work?*

10.9.5 Observations

fig. 15 and fig. 16 shows the following:

- Interactive workloads benefits from network CPU cost savings when using the SMC-Rv2 protocol. The network CPU cost savings become significant as the payload size increases. The network CPU cost savings is an outcome of not performing the typical TCP processing of packets (e.g., segmentation, flow control, congestion control, etc.) in the stack.
- Bulk transfer workloads benefits from significant network CPU cost savings when using the SMC-Rv2 protocol. The reasoning is the same as above. In both streaming workloads (i.e., GET & PUT), the receiver, who is responsible for processing a large payload (e.g., 20MB), has greater network CPU cost savings as all of that work is offloaded to the network interface controller (NIC).

11 Hardware Performance: z17

11.1 HiperSockets: z17 vs z16

11.1.1 Background

HiperSockets is a hardware feature that provides high speed LPAR to LPAR communication within the same CPC. It is a processor to memory architecture rather than processor to I/O. Due to the intra-traffic characteristic, the following perks are offered at a higher level: network availability, security, simplicity, performance, and cost reduction [18].

11.1.2 Testing Environment: HiperSockets on z17

The goal of this test was to benchmark HiperSockets performance on the latest mainframe generation (e.g., z17) then compare it against previous generation (e.g., z16).

11.1.3 z/OS Environment Configuration: HiperSockets on z17

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17 & z16
 - Machine Type (Model): 9175 - ME1
 - Machine Type (Model): 3931 - A01
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- Interfaces
 - HiperSockets Maximum Frame Size: 16KB
 - TCP/IP MTU Size: 8KB

11.1.4 Results

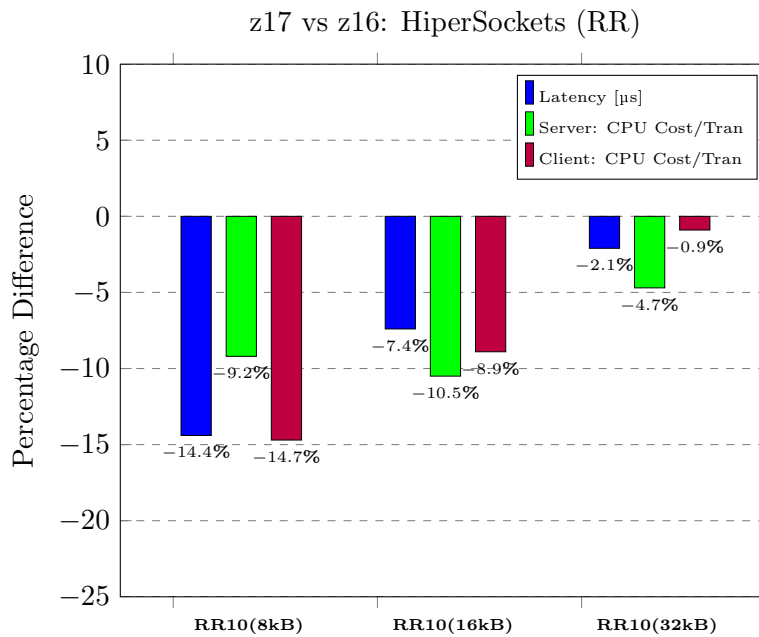


Figure 17: This graph answers the following question: *How much faster and more efficient in using system resources (CPU) is HiperSockets on z17 compared to z16 for interactive workloads?*

11.1.5 Observations

In reference to fig. 17, the following conclusion can be drawn:

- HiperSockets on z17 offers better performance for interactive workloads across all metrics such as network latency and network CPU cost

11.2 SMC-Dv2: z17 vs z16

11.2.1 Background

Shared Memory Communications Direct Memory Access (SMC-D) uses Internal Shared Memory (ISM) in allowing two SMC capable peers to communicate intra-CPC. During the TCP connection handshake, the capable peers dynamically detect SMC eligibility before using it. The protocol boosts workload performance by providing a low latency and high bandwidth solution. Refer to [19] for more information.

11.2.2 Testing Environment: SMC-Dv2 on z17

The goal of this test was to benchmark SMC-Dv2 performance on the latest mainframe generation (e.g., z17) then compare it against the previous generation (e.g., z16).

11.2.3 z/OS Environment Configuration: SMC-Dv2 on z17

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17 & z16
 - Machine Type (Model): 9175 - ME1
 - Machine Type (Model): 3931 - A01
- Release: 3.2

- Number of CPUs: 4 (Dedicated) per LPAR
- Interface: ISMv2

11.2.4 Results

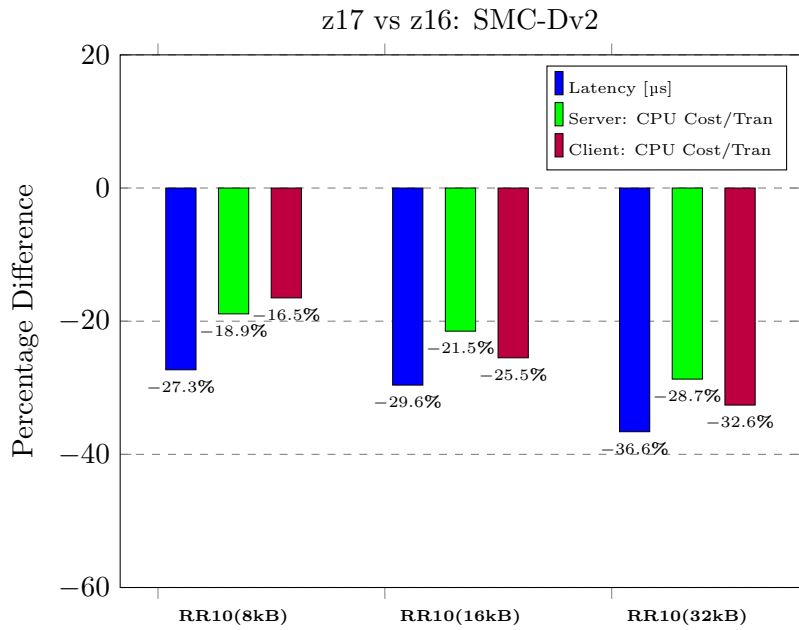


Figure 18: This graph answers the following question: *How much faster and more efficient in using system resources (CPU) is SMC-Dv2 on z17 compared to z16 for interactive workloads?*

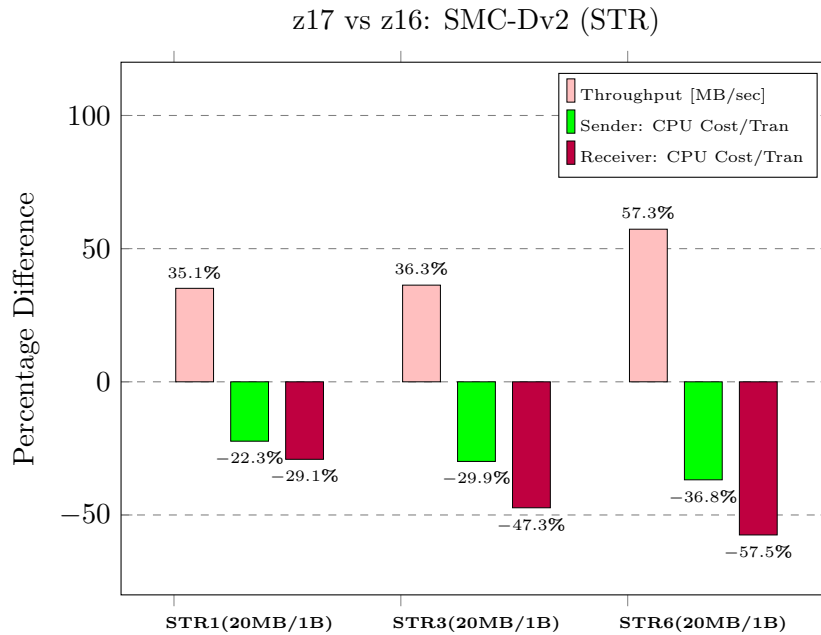


Figure 19: This graph answers the following question: *How much faster and more efficient in using system resources (CPU) is SMC-Dv2 on z17 compared to z16 for streaming workloads?*

11.2.5 Observations

In reference to fig. 18 and fig. 19, the following conclusions can be drawn:

- SMC-Dv2 handling interactive workload on z17 offers less networking latency and less networking CPU cost for the client & server when compared to z16
- SMC-Dv2 handling streaming workload on z17 offers more throughput and less networking CPU cost for the sender & receiver when compared to z16

11.3 z17: HiperSockets vs SMC-Dv2

11.3.1 Background

In the previous sections, the HiperSockets and SMC-Dv2 protocol were explained. In this section, both of these protocols will be compared against each other when running on z17.

11.3.2 Testing Environment: HiperSockets vs SMC-Dv2

The goal of this test was to compare SMC-Dv2 against HiperSockets on z17.

11.3.3 z/OS Environment Configuration: HiperSockets vs SMC-Dv2

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
Machine Type (Model): 9175 - ME1
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- Interfaces:
ISMv2
HiperSockets Maximum Frame Size: 16KB
TCP/IP MTU Size: 8KB

11.3.4 Results

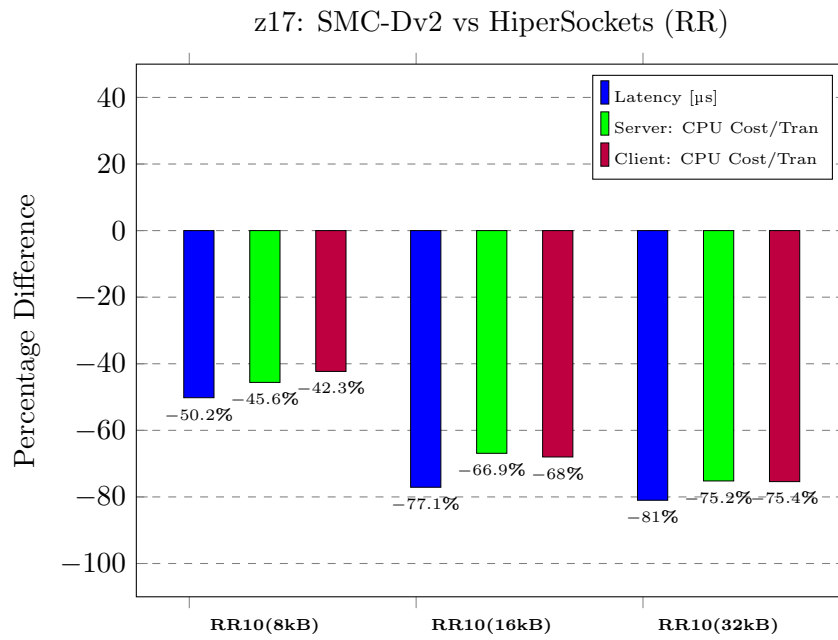


Figure 20: This graph compares SMC-Dv2 against HiperSockets on z17. This graph answers the following question: *How much faster and cheaper is SMC-Dv2 compared to HiperSockets running on z17 for interactive workloads?*

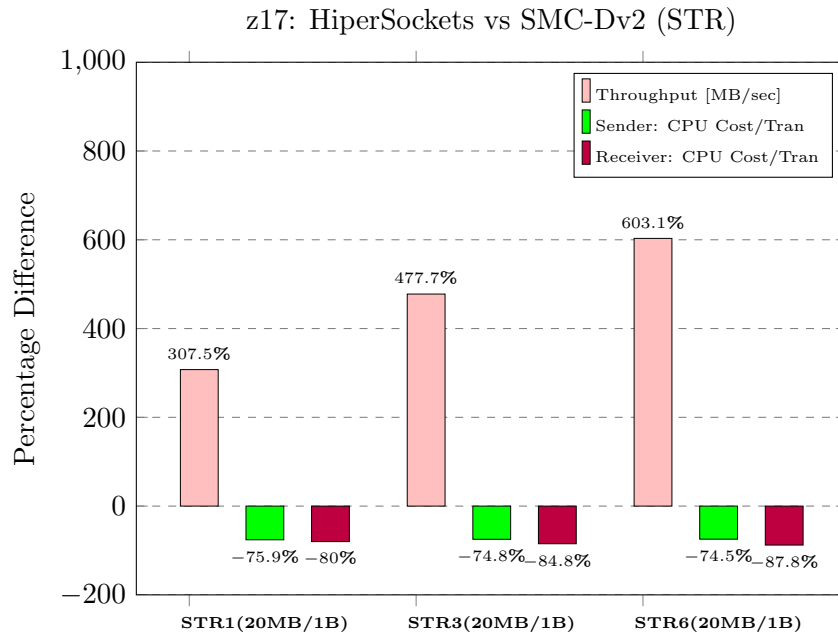


Figure 21: This graph compares SMC-Dv2 against HiperSockets on z17. This graph answers the following question: *How much faster and cheaper is SMC-Dv2 compared to HiperSockets running on z17 for streaming workloads?*

11.3.5 Observations

In reference to fig. 20 and fig. 21, the following conclusion can be drawn:

- SMC-Dv2 handling interactive workload on z17 offers less network latency and reduced networking CPU cost for the client & server when compared to HiperSockets
- SMC-Dv2 handling streaming workload on z17 offers more throughput and less network CPU cost for the sender & receiver when compared to HiperSockets

12 Network Security: AT-TLS

12.1 AT-TLS: Legacy RSA Signature Pair Against RSASSA-PSS Signature Pair

12.1.1 Background

Application Transparent Transport Layer Security (AT-TLS) is a function within the TCP/IP stack that allows for transparent implementation of TLS protection to TCP traffic through policies. The optimized integration between System SSL and AT-TLS ensures no extra overhead in the AT-TLS path versus direct calls to System SSL.

12.1.2 Testing Environment: Legacy RSA Signature Pair Against RSASSA-PSS Signature Pair

fig. 22 shows our AT-TLS testing environment set-up.

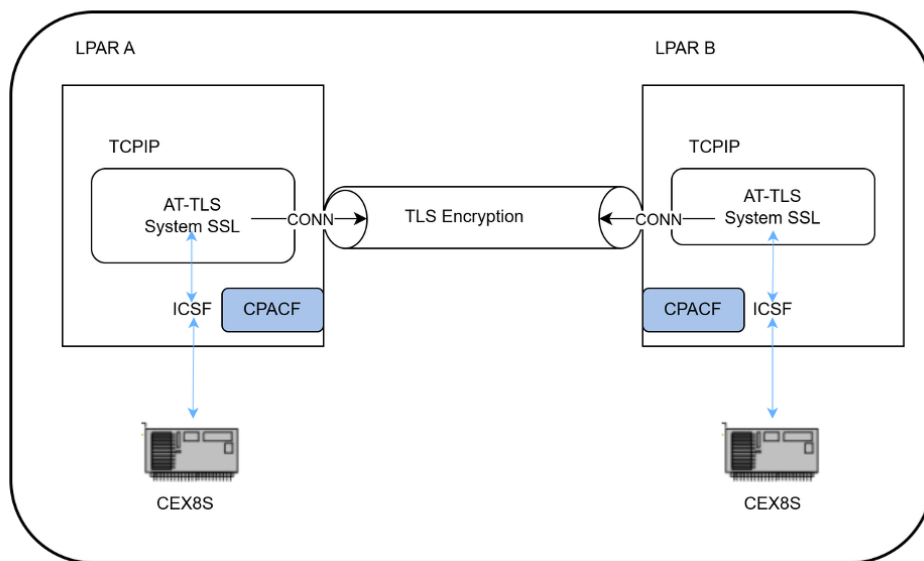


Figure 22: AT-TLS testing environment

In this test, the comparison consisted of using different signature pairs against the TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (C030) cipher. The TLS_SIGALG_SHA256_WITH_RSASSA_PSS (0804) signature pair was compared to the TLS_SIGALG_SHA256_WITH_RSA (0401) signature pair running under the TLSv1.2 security protocol.

12.1.3 z/OS Environment Configuration: Legacy RSA Signature Pair Against RSASSA-PSS Signature Pair

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- System SSL Version 5.1
Service Level: OA66110
- ICSF Level: HCR77F0
- The Crypto Express8 (CEX8S) adapter is configured as CCA coprocessor

- Cipher Suite
 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (C030)
- Signature Pairs
 TLS_SIGALG_SHA256_WITH_RSA (0401)
 TLS_SIGALG_SHA256_WITH_RSASSA_PSS (0804)
- Workload: CRR40(64B/8kB)

12.1.4 Results

RSA PKCS#1 v1.5 versus RSASSA-PSS Signature Pair

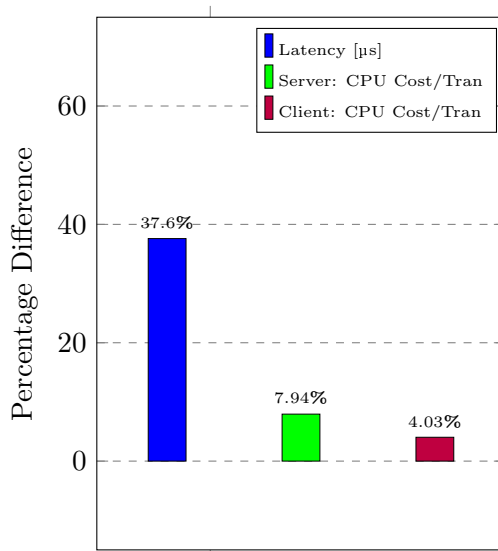


Figure 23: This graph compares RSASSA-PSS signature pair against RSA PKCS#1 v1.5 to understand what is the performance impact of using RSASSA-PSS signature pair in terms of network latency and client & server network CPU cost.

12.1.5 Observations

In reference to fig. 23, the following conclusion can be drawn:

- An increase in network latency and network CPU cost (client and server) will be observed when using the newly more secured RSASSA-PSS signature pair instead of the legacy RSA on a TLSv1.2 cipher

12.2 AT-TLS: AES-CBC versus AES-GCM

12.2.1 Background

AES-CBC and AES-GCM are both symmetric encryption algorithms used for payload encryption. AES in CBC mode uses SHA-384 HMACs for data integrity. AES in GCM mode does payload encryption and data integrity together.

12.2.2 Testing Environment: AES-CBC versus AES-GCM

fig. 22 shows the testing environment. In this test, the goal was to benchmark AES-CBC against AES-GCM. The reader should note the following:

- For AES-CBC, System SSL uses CPACF directly.
- For AES-GCM, System SSL calls ICSF which invokes CPACF on System SSL behalf.

12.2.3 z/OS Environment Configuration: AES-CBC versus AES-GCM

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- System SSL Version 5.1
 - Service Level: OA66110
- ICSF Level: HCR77F0
- The Crypto Express8 (CEX8S) adapter is configured as CCA coprocessor
- Cipher Suite
 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (C030)
 - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (C028)
- Workload: CRR40(64B/8kB)

12.2.4 Results

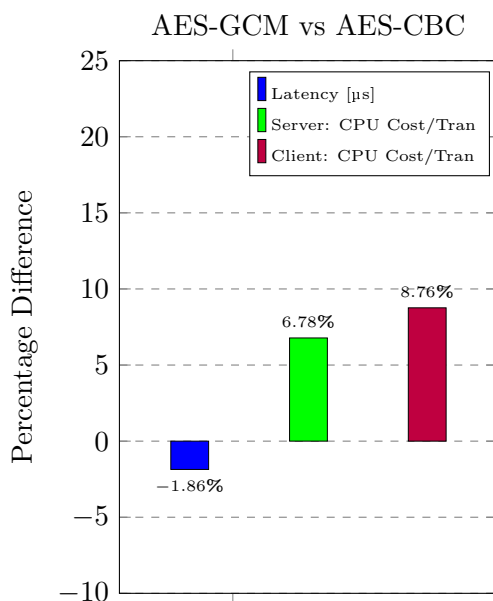


Figure 24: This graph compares AES-GCM against AES-CBC in understanding how much more costly is using AES-GCM which provides payload encryption and data integrity together

12.2.5 Observations

In reference to fig. 24, the following conclusion can be drawn:

- AES-GCM provides a more secure data encryption at a similar cost as AES-CBC with SHA-384 HMAC

It also does not degrade network latency

12.3 AT-TLS: TLSv1.3 vs TLSv1.2

12.3.1 Background

In general, TLSv1.3 is more secure at the cost of being more computational heavy compared to TLSv1.2. It has multiple key derivation operations per handshake and makes ephemeral key agreement (e.g., finite field DHE or Elliptic Curve ECDHE) mandatory for all standard connections.

12.3.2 Testing Environment: TLSv1.3 vs TLSv1.2

fig. 22 shows the testing environment. In this test, the goal was to benchmark TLSv1.3 against TLSv1.2. The reader should note the following:

- Both cipher suites uses ephemeral elliptic curve Diffie-Hellman (ECDHE) for key exchange mechanism
- Both security protocol uses RSASSA-PSS for signature pair operations and AES-GCM for payload encryption & data integrity

12.3.3 z/OS Environment Configuration: TLSv1.3 vs TLSv1.2

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- System SSL Version 5.1
Service Level: OA66110
- ICSF Level: HCR77F0
- The Crypto Express8 (CEX8S) adapter is configured as CCA coprocessor
- Cipher Suites
TLSv1.2: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (C030)
TLSv1.3: TLS_AES_256_GCM_SHA384 (1302)
- Workload: CRR40(64B/8kB)

12.3.4 Results

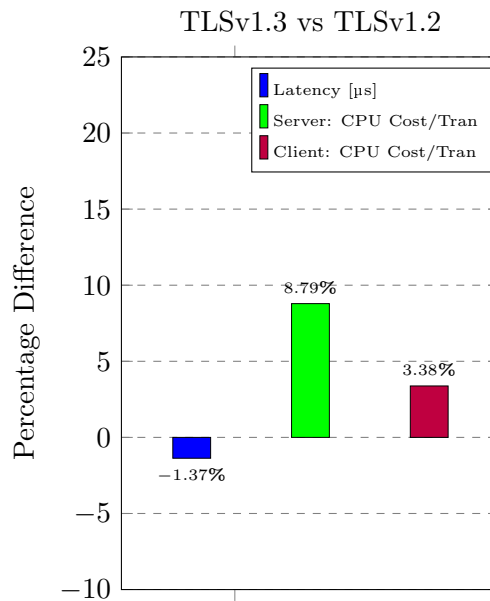


Figure 25: This graph compares the network latency and network CPU cost of running the TLSv1.3 protocol against the TLSv1.2 protocol on the latest mainframe (e.g., z17)

12.3.5 Observations

In reference to fig. 25, the following conclusions can be drawn:

- TLSv1.3 incurs a little increase in network CPU usage while offering the network response times as TLSv1.2 when using similar algorithms on z17

13 TLS Performance Paper

A TLS performance paper was published by the z/OS Communications Server Performance Team. It focuses on AT-TLS performance using TLSv1.2 and TLSv1.3. The new paper, z/OS Communications Server TLS Performance Update, is accessible at [\[20\]](#).

14 ICSF Optimization

14.0.1 Background

Integrated Cryptographic Service Facility (ICSF) options AUDITKEYLIFETKDS and AUDITPKCS11USG provide a set of options that control auditing lifecycle and events related to PKCS #11 services. Setting SESSIONOBJ to YES audits the CREATE, USE, or DELETE of a key that exists only for a single TLS session. SESSIONOBJ is intended for ephemeral keys and is very expensive. Also, SESSIONOBJ does not provide any useful data hence the recommendation is to set TOKENOBJ to YES and SESSIONOBJ to NO.

14.0.2 Testing Environment: TOKENOBJ(YES) & SESSIONOBJ(NO)

The metrics were gathered under the following options:

- Security Protocol: TLSv1.3
- Cipher Suite: TLS_AES_128_GCM_SHA256 (1301)

- Audit Options
 AUDITKEYLIFETKDS
 AUDITPKCS11USG

14.0.3 z/OS Environment Configuration: TOKOBY(YES) & SESSOBY(NO)

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2
- Number of CPUs: 4 (Dedicated) per LPAR
- System SSL Version 5.1
 Service Level: OA66110
- ICSF Level: HCR77F0
- The Crypto Express8 (CEX8S) adapter is configured as CCA coprocessor
- Cipher Suites
 TLSv1.3: TLS_AES_128_GCM_SHA256 (1301)
- Workload: CRR40(64B/8kB)

Impact Of Setting TOKOBY(YES) & SESSOBY(YES)

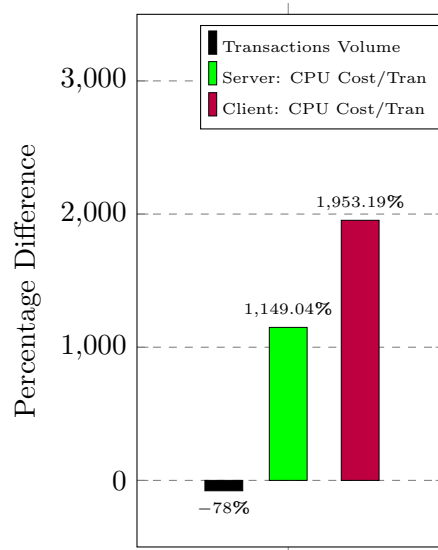


Figure 26: This graph compares the transactions volume and network CPU cost (client and server) when TOKOBY(YES) SESSOBY(YES) is configured versus TOKOBY(YES) SESSOBY(NO). In other words, the graph answers the following question: *What is the cost of turning on SESSOBY?*

14.0.4 Observations

To recap with reference to fig. 26, configuring the settings of TOKOBJ(YES) SESSOBJ(YES) instead of TOKOBJ(YES) SESSOBJ(NO) has the following impact:

- The transaction volume decreases by 78% when TOKOBJ(YES) SESSOBJ(YES) used instead of TOKOBJ(YES) SESSOBJ(NO)
- The server networking CPU cost increases by 1149% when TOKOBJ(YES) SESSOBJ(YES) used instead of TOKOBJ(YES) SESSOBJ(NO)
- The client networking CPU cost increases by 1953% when TOKOBJ(YES) SESSOBJ(YES) used instead of TOKOBJ(YES) SESSOBJ(NO)

15 3.2 versus 3.1: Release To Release Comparison

15.1 z/OS Communications Server 3.2 versus 3.1

15.1.1 Introduction

In this sub-section, the pure focus was benchmarking the latest release (3.2) against the previous release (3.1) on the latest CPC model (e.g., z17)

15.1.2 Testing Environment: z/OS Communications Server 3.2 versus 3.1

The operating system (OS) version comparison was done on the latest CPC (e.g., z17). On z17, the OS supports OSA-Express feature and Network Express feature. Therefore, both NIC types were tested.

15.1.3 z/OS Environment Configuration: z/OS Communications Server 3.2 versus 3.1

Below is the environment configuration in which the data was collected:

- Central Processor Complex (CPC): z17
- Release: 3.2 & 3.1
- Number of CPUs: 4 (Dedicated) per LPAR
- Interfaces
 - OSA-Express 7S CHPID Type OSD 25GbE
 - Network Express CHPID Type OSH 25GbE
- Frame Size
 - Regular Frames
 - Jumbo Frames
- Workloads
 - RR60(4kB)
 - CRR40(64B/8kB)
 - STR3(1B/20MB)
 - STR3(20MB/1B)

15.1.4 Observations

The below bullets summarizes the testing results:

- z/OS 3.2 running all the listed workloads over **OSD** on z17 with regular & jumbo frames performance is *comparable to* z/OS 3.1 running all the listed workloads over **OSD** on z17 with regular & jumbo frames
- z/OS 3.2 running all the listed workloads over **OSH** on z17 with regular & jumbo frames performance is *comparable to* z/OS 3.1 running all the listed workloads over **OSH** on z17 with regular & jumbo frames

16 z/OS Communications Server Performance Index

The following URL, <https://www.ibm.com/support/pages/node/317829>, contains all z/OS Communications Server Performance related publications. The posted materials are updated as necessary.

References

- [1] J. Stevens, “IBM z/OS Communications Server and OSA-Express Best Practices,” Technical Report, IBM, Raleigh, USA, August 2023.
- [2] “Abstract for Resource Measurement Facility User’s Guide,” April 2025. Accessed: Jan 28, 2026.
- [3] “INTERFACE - IPAQENET OSA-Express QDIO interfaces statement,” September 2025. Accessed: Jan 28, 2026.
- [4] “Inbound workload queuing,” September 2025. Accessed: Jan 28, 2026.
- [5] D. Herr, “Getting the most out of your OSA (Open Systems Adapter) with z/OS Comm Server,” 2013.
- [6] B. White, O. Ferreira, T. Missawa, and T. Sudewo, *IBM z/OS V2R2 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*. USA: IBM, 2016.
- [7] “QDIO Accelerator,” September 2025. Accessed: Jan 29, 2026.
- [8] “HEAPPOOLS64 (C/C++ and AMODE 64 only),” September 2025. Accessed: Jan 29, 2026.
- [9] K. Rashad, “V2R5: z/OS Communications Server Performance Summary Report,” Technical Report, IBM, Raleigh, USA, May 2022.
- [10] S. Guendert, “Understanding High Performance FICON (zHPF) Part 1: Protocol specifics,” Technical Report, Brocade, 2013.
- [11] K. Rashad, “3.1: z/OS Communications Server Performance Summary Report,” Technical Report, IBM, Raleigh, USA, April 2025.
- [12] “z/OS Communications Server support for the Network Express feature on IBM z17),” September 2025. Accessed: Feb 3, 2026.
- [13] “Network Express feature in EQDIO mode,” September 2025. Accessed: Feb 3, 2026.
- [14] “z/OS Communications Server use of Network Express Feature,” November 2025. Accessed: Feb 3, 2026.
- [15] “IBM z/OS Communications Server and OSA-Express Best Practices,” August 2023. Accessed: Feb 4, 2026.
- [16] “Enterprise Extender,” 2010. Accessed: Feb 4, 2026.
- [17] “Inbound workload queuing,” September 2025. Accessed: Feb 9, 2026.
- [18] “What is a HiperSocket?,” December 2019. Accessed: Feb 4, 2026.
- [19] “Shared Memory Communications - Direct Memory Access,” April 2025. Accessed: Feb 5, 2026.
- [20] C. Nyamful, “z/OS Communications Server TLS Performance Update,” Technical Report, IBM, Raleigh, USA, February 2024.