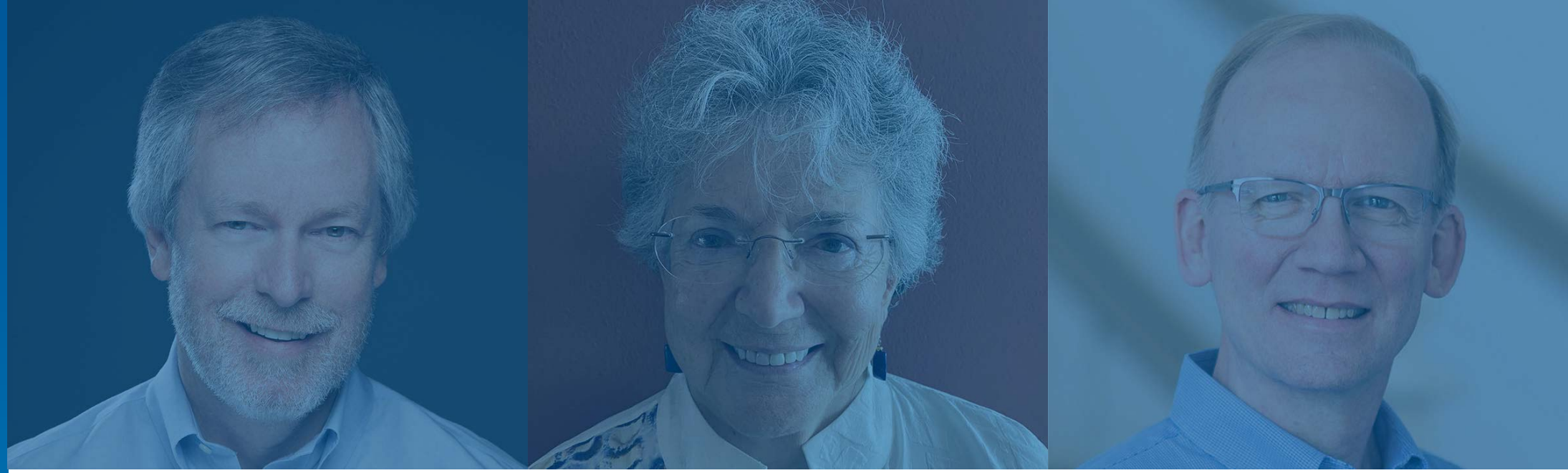


IntelliMagic zAcademy Session #39

Insights into New XCF Path Usage Metrics

Frank Kyne, Watson & Walker
Todd Havekost, IntelliMagic

October 25, 2022

Availability Intelligence

Agenda

- Introduction to XCF Signaling and Transport Class Simplification
- Managing MAXMSG Values
- New Path Usage Concepts
- Analyzing New Path Usage Metrics
- Summary / Resources / Questions

Introduction to XCF Signaling and Transport Class Simplification

XCF Basics

- XCF provides high speed resilient messaging between peer programs running on the same or different systems in the sysplex. We generally find 50-100 exploiters per system.
- XCF typically uses CF structures to provide paths from each system to every other system in the sysplex, with buffers at each end of each path to ensure smooth and timely flow of messages.
- **A key concept in XCF is that, from XCF's perspective, signaling paths are point-to-point and uni-directional.**
  - For example, even though a CF structure enables every-to-every 2-way communication, you still define PATHIN and PATHOUT separately in your COUPLExx member of Parmlib.
  - And all XCF signaling reporting is on the basis of activity between a *pair of systems.*

Intro to Transport Class Simplification

- In z/OS 2.4, IBM delivered enhancements to XCF called Transport Class Simplification (TCS).
  - This enhancement is ENABLEd by default, and no changes to your definitions or infrastructure are required. ✓
  - The implementation was *so effortless* that many customers didn't even notice the change. ✓

- To understand how TCS 'Simplified' things, let's have a quick look at how XCF signaling worked *before* TCS, and how it works *now*.

Life Before TCS

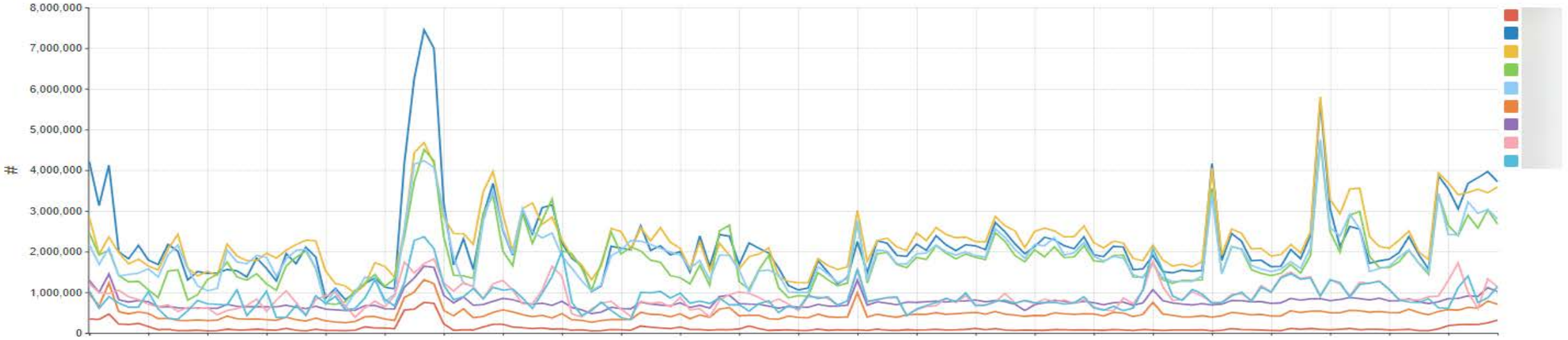
- Because XCF supports message sizes from <1 KB up to 61 KB, and CPC memory was MUCH smaller when XCF was introduced (c. 1990), XCF provides a way to group paths (‘transport classes’), and optimize the size of each buffer for each transport class (‘CLASSLEN’).
  - It also lets you control the maximum amount of memory that can be used by each path’s buffer *pool* (MAXMSG).
- XCF generally used the message size to assign each message to a specific transport class.

Life Before TCS

- Because signaling resources were divided into multiple transport classes, the system programmer (in theory) had to:
  - Identify the volume of messages of each size.
    - Not easy because the only way to see the number of each sized message is using D XCF,CD,CLASS=ALL commands (this info is not in SMF).
  - Take into account that message volumes and mix of sizes varied by time, day, and system.
    - This info IS in SMF, however it is reported separately for each pair of systems – there are no XCF sysplex-level SMF records.
  - With that info, identify how many paths and buffers were needed *for each subset*.
  - Then define that configuration in the COUPLExx Parmlib member.

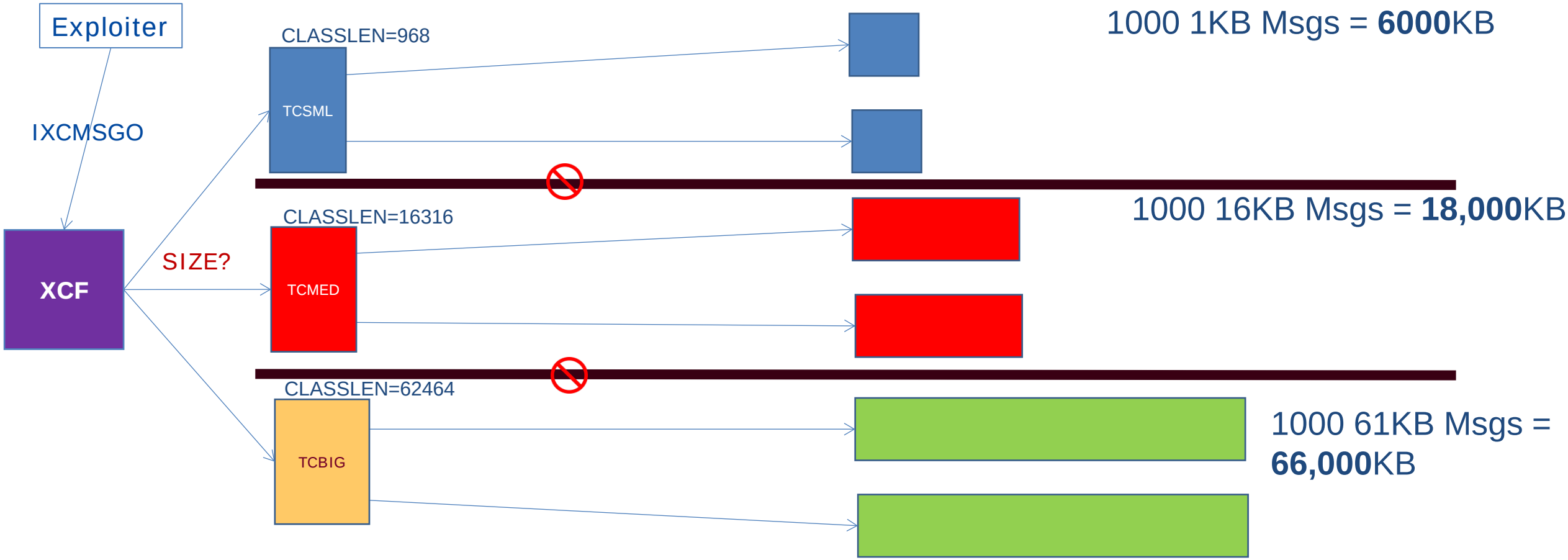
Life Before TCS

- This is a very time-consuming exercise.
  - It is probably impossible to have *static* definitions that were always perfect for *every* system in a *constantly-changing* environment.



And because of the number, format, and inter-relationship between the definition statements, it is not unusual to find definition mistakes.

Life Before TCS (2-way sysplex)

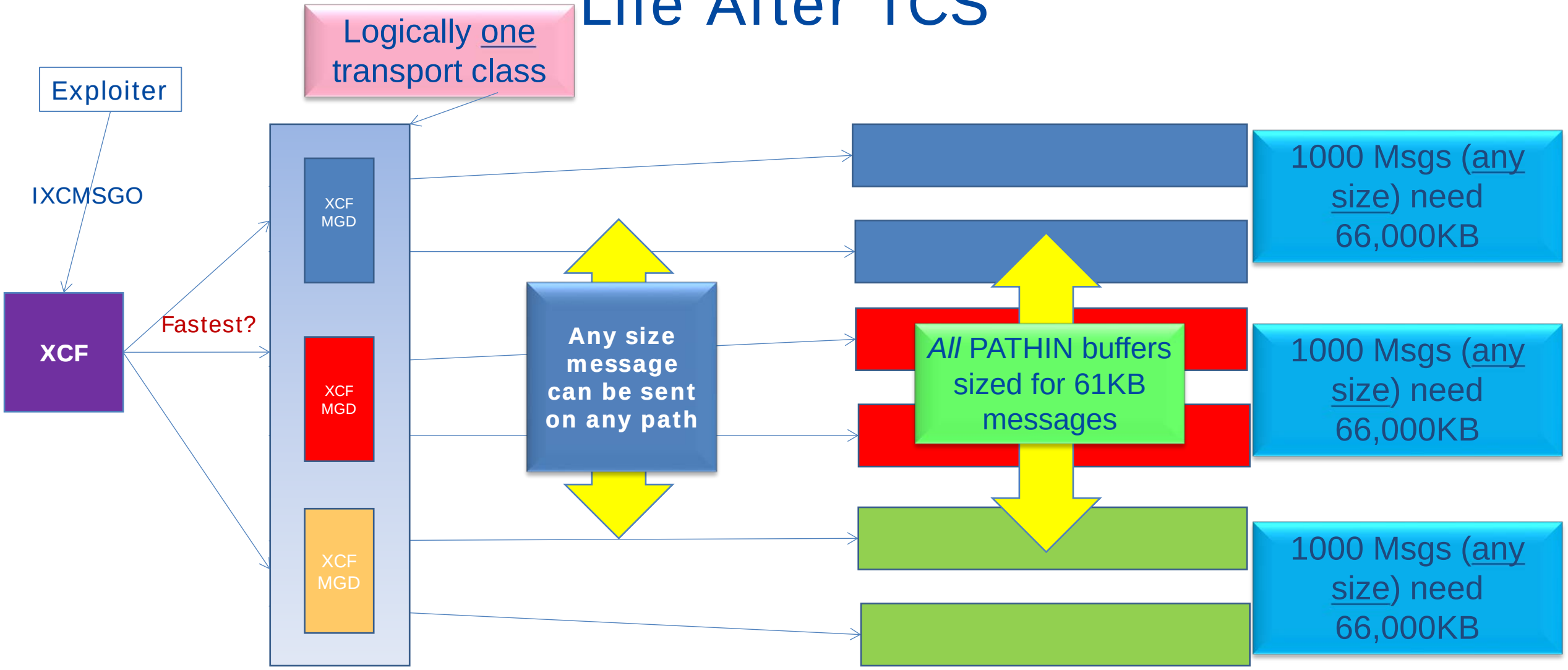


Every buffer consists of 1 or more 4KB pages, plus 2KB of control information

How Transport Class Simplification Helps

- The ‘Transport Class Simplification’ enhancement is intended to let XCF pool your signaling resources, using your existing definitions, and manage them dynamically.
  - Rather than certain message sizes being limited to a given transport class and its associated resources, any XCF message can potentially be sent on *any* XCF path.
    - This significantly increases the amount of storage that is available for messages of a given size.

Life After TCS



Life After TCS

- Because any message can potentially use any path, you no longer need to worry about the breakout of message sizes.
- All you need to do is to ensure XCF has enough paths and buffers to handle the overall messaging workload, and then leave it to XCF to manage those resources.
  - Because of the traditional method of dividing XCF signaling resources across multiple transport classes, each sized to handle its own peak traffic, most sysplexes have more paths and buffers than they really need.

Life After TCS

- This presents opportunities for optimization – Todd will be describing how the new XCF RMF metrics can help you with that.
- However, it also increases the possibility of excessive storage consumption in extreme circumstances in systems with huge MAXMSG values.

Managing MAXMSG Values

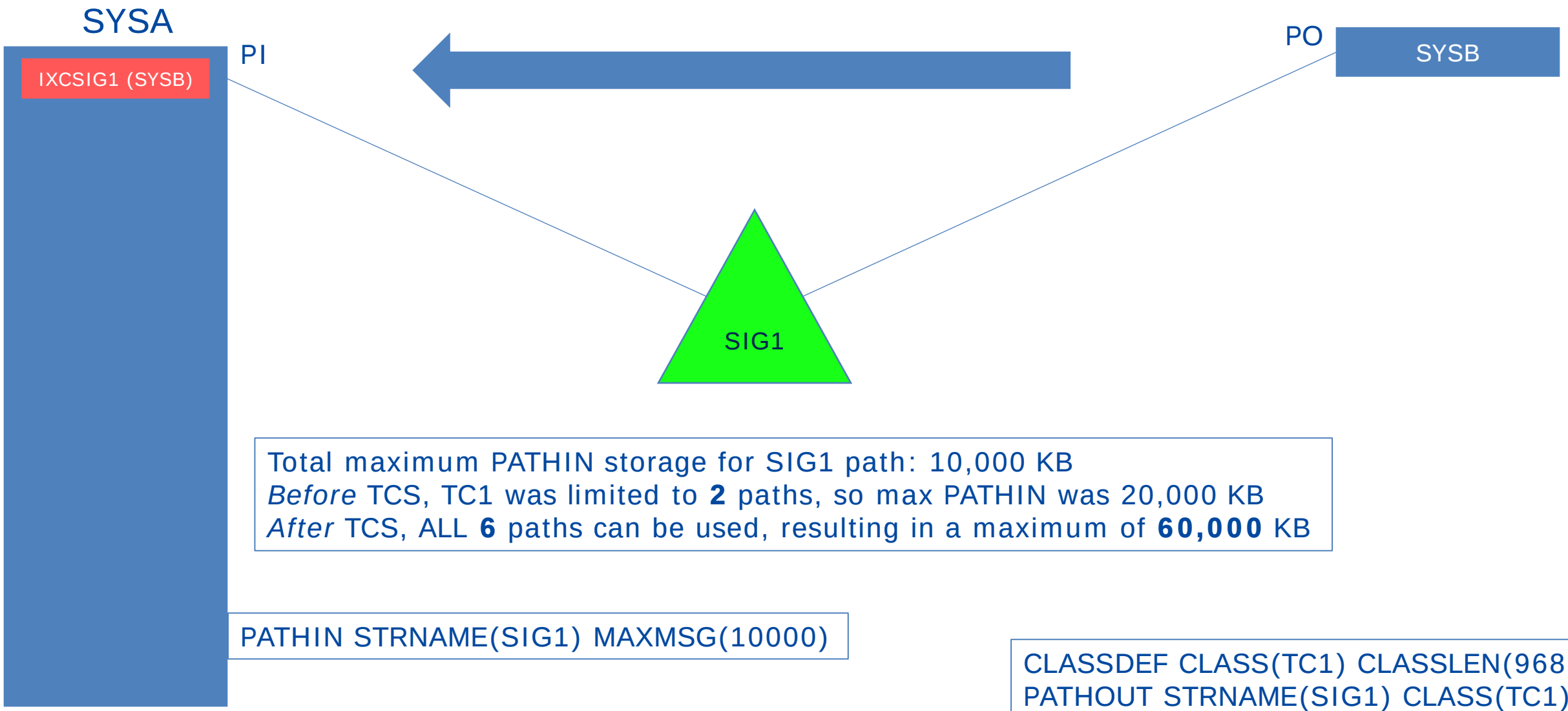
Finding Appropriate MAXMSG Value

- Traditionally, it was considered a ‘bad thing’ if XCF ran out of buffers (reported in an RMF PP XCF Path Activity report as ‘BUFFERS UNAVAIL’):
  - The normal response to this was to increase the maximum size (MAXMSG) of the buffer pool for that PATHIN.
- However, incremental improvements to XCF over the years have reduced the seriousness of that situation, meaning that very large buffer pools generally are *not* required.
- Also, the *real* reason for trying to minimize the number of times XCF ran out of buffers was to avoid a performance impact to XCF exploiters.
  - However, there was no way to accurately identify that impact, so the only available course of action was to increase the MAXMSG.
- Let’s see how TCS and related new metrics have changed this.

Finding Appropriate MAXMSG Value

- As we saw, when TCS is enabled, *all* incoming messages now reside in 64KB buffers.
  - For small messages, which typically make up about 90% of messages, this is an 11x increase in storage for each message.
- Normally, messages arrive and are retrieved by the target address space very quickly, so the extra few KB are irrelevant.
- But what happens if the target address space is slow or stopped or just can’t keep up with the message arrival rate?

2-Way Sysplex

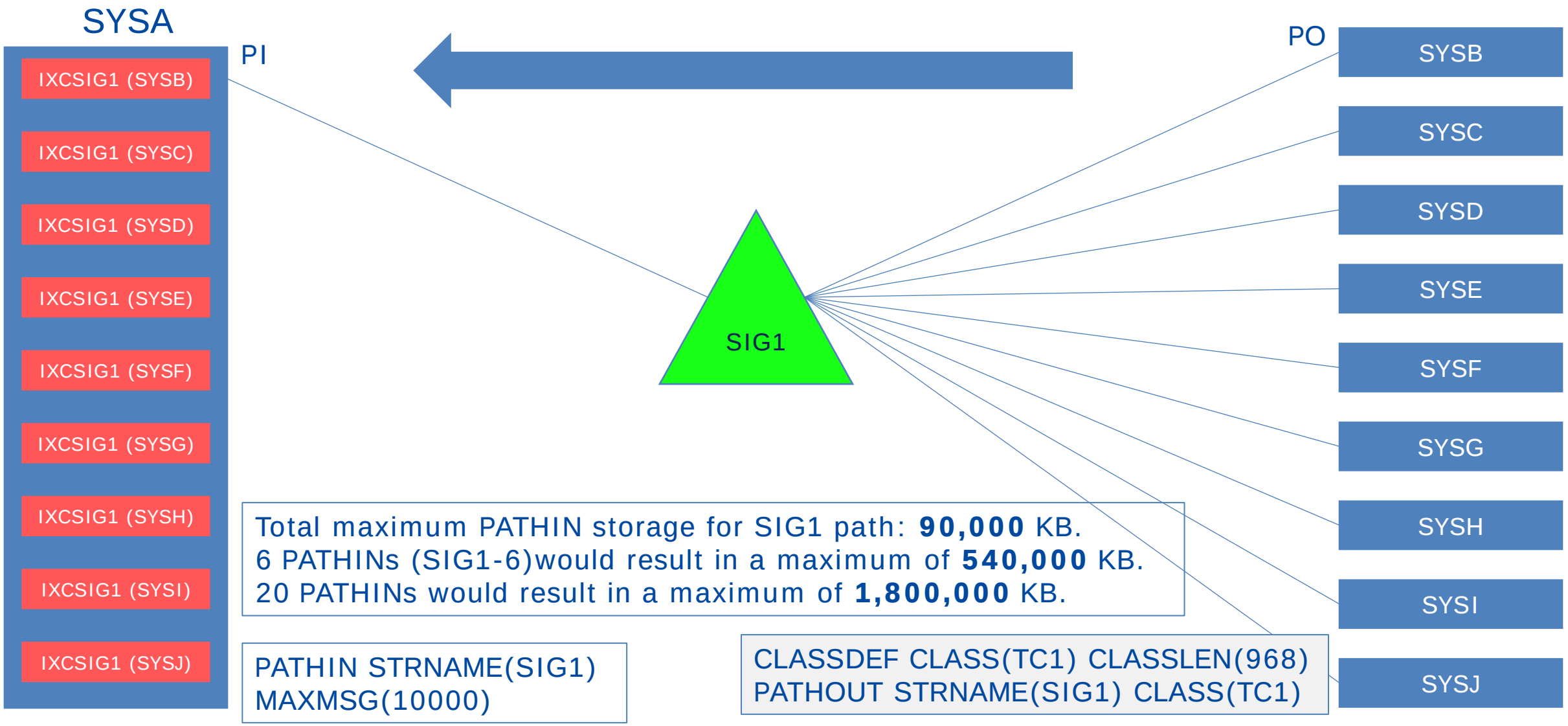


Total maximum PATHIN storage for SIG1 path: 10,000 KB
Before TCS, TC1 was limited to **2** paths, so max PATHIN was 20,000 KB
After TCS, ALL **6** paths can be used, resulting in a maximum of **60,000** KB

PATHIN STRNAME(SIG1) MAXMSG(10000)

CLASSDEF CLASS(TC1) CLASSLEN(968)
PATHOUT STRNAME(SIG1) CLASS(TC1)

10-way Sysplex



Finding Appropriate MAXMSG Value

- Why do we care?
- Incoming *and* outgoing XCF messages (*plus* other XCF control information) reside in a 2GB XCF data space while waiting to be retrieved or sent.
- If there is a flood of messages *and* the maximum buffer pool sizes are very large *and* messages are not being moved out of the buffers, it is possible for XCF to consume all the storage in its data space – this currently results in a 0A2-040 wait state (see open HIPER APAR OA62980).
  - *Additionally*, if XTCSIZE is enabled, the PATHIN buffers that used to reside in 31-bit real storage, are now placed in above the bar 64-bit real. This was delivered by HIPER APAR OA60480.
- To protect systems from this risk, IBM issued the following guidance:
  - Aim to have PATHIN MAXMSG values not greater than 2000.
  - *Total* PATHIN MAXMSG values should be < 800,000.

Finding Appropriate MAXMSG Value

- How to determine *your* Total PATHIN MAXMSG value?
- Fastest way is to use D XCF,PI,STRNM=ALL command:

```
D XCF,PI,STRNM=ALL
IXC356I  15.53.50  DISPLAY XCF 071
STRNAME          REMOTE   PATHIN      UNUSED                    LAST  MXFER
PATHIN           SYSTEM   STATUS      PATHS   RETRY  MAXMSG  RECVD TIME
IXC_DEFAULT_1             WORKING        10      10    2000      -     -
                 FPK1     WORKING                                4959   135
IXC_DEFAULT_2             WORKING        10      10    2000      -     -
                 FPK1     WORKING                                37561   71
IXC_DEFAULT_3             WORKING        10      10    2000      -     -
                 FPK1     WORKING                                14754  154
IXC_DEFAULT_4             WORKING        10      10    2000      -     -
                 FPK1     WORKING                                49231   95


STRNAME          REMOTE   PATHIN      DELIVRY BUFFER  MSGBUF SIGNL
PATHIN    LIST   SYSTEM   STATUS      PENDING LENGTH  IN USE NUMBR NOBUF
IXC_DEFAULT_1
             9   FPK1     WORKING           0  62464       0  4959     2
IXC_DEFAULT_2
             9   FPK1     WORKING           0  62464      66 37561     0
IXC_DEFAULT_3
             9   FPK1     WORKING           0  62464       0 14754     2
IXC_DEFAULT_4
             9   FPK1     WORKING           0  62464       0 49231     0
```

Make sure you get this info for EVERY connected system.

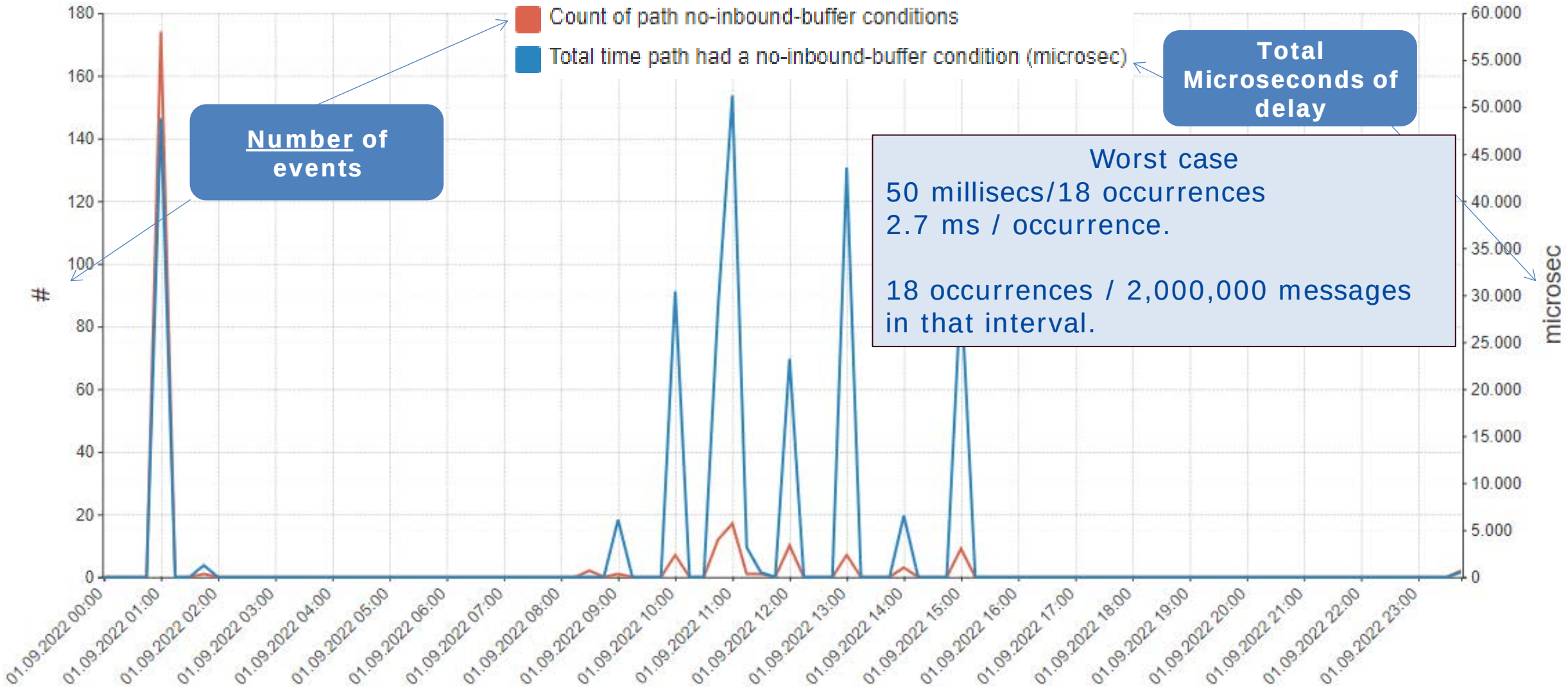
Finding Appropriate MAXMSG Value

- What to do if you find your PATHIN MAXMSGs add up to much more than 800,000?
  - If *in*creasing MAXMSG was the traditional way to address No Buffer conditions, will *de*creasing MAXMSG not increase # of No Buffer?

- Maybe, but will anyone notice? Luckily, the new metrics have that covered! There are two new fields in the 74.2 SMF record:
  - Count of number of no buffer conditions that actually resulted in a message being delayed.
  - Total delay time for those messages.

Impactful No-Inbound-Buffer Conditions

For System ID ‘SYS1’ with Target System ID ‘SYS2’



Optimizing MAXMSG values

- If you want/need to reduce your MAXMSG values:
  - Check your *current* Impactful No Buffer counts and times.
  - You can adjust MAXMSG values dynamically – SETXCF MODIFY,PI,STRNM=whatever,MAXMSG=some_smaller_value
    - The scope of this command is a *single system*. It needs to be issued on every system where you want to make the change.
  - Keep an eye on the Impactful No Buffer values as you go along.
  - It is *not* necessary to adjust the PO and PI values at the same time – we recommend leaving PO MAXMSG values as they are until you are finished adjusting PI MAXMSG values.

Managing MAXMSG

IBM's experience has been that excessive memory usage on the *sending* side (transport class and PATHOUT) is very uncommon.

- It is very unlikely that every XCF exploiter on a given system will suddenly spring to life and bombard their peers on every other system.
- The exception is if a system dies but is not partitioned from the plex in a reasonable time – XCF will continue accepting messages for that system until it is removed from the sysplex, but is unable to send them – so they will starting queueing in the PATHOUT buffer pools.
  - **Recommendation**: Follow IBM Best Practice and ensure that BCPii and SSDPP are enabled.

Summary

TCS is great – don't disable it!

Apply the PTF for OA60480 if not already applied.

Subscribe to HIPER APAR OA62980 and apply when available.

Aim to get the sum of PATHIN MAXMSGs under 800,000.

In normal running, keep an eye on Impactful No Buffer times.

- Let us know if they become 'uncomfortable'.

Unrelated but important APAR:

If you duplex your lock structures, check HIPER APAR OA63312 and apply the PTFs (available now).

New XCF Path Usage Concepts

Rationale for New Path Usage Metrics

- With TCS no longer need to manage transport classes
- New metrics designed to inform remaining configuration items
  - Number of paths
  - Number of buffers
- Prerequisites that have not changed
  - Good performance on all systems across the sysplex processing messages
  - Good performance on signaling paths (typically CF list structures in today's environments)

XCF Path Utilization

- Measured on inbound side
- Signals are read in parallel in “buckets” of up to 4 signals
- XCF reports “utilizations” as discrete values of
  - 25% - 1 read active
  - 50% - 2 reads active
  - 75% - 3 reads active
  - 100% - 4 reads active
- A bucket ends when the I/O completes for all “N” signals in the current bucket

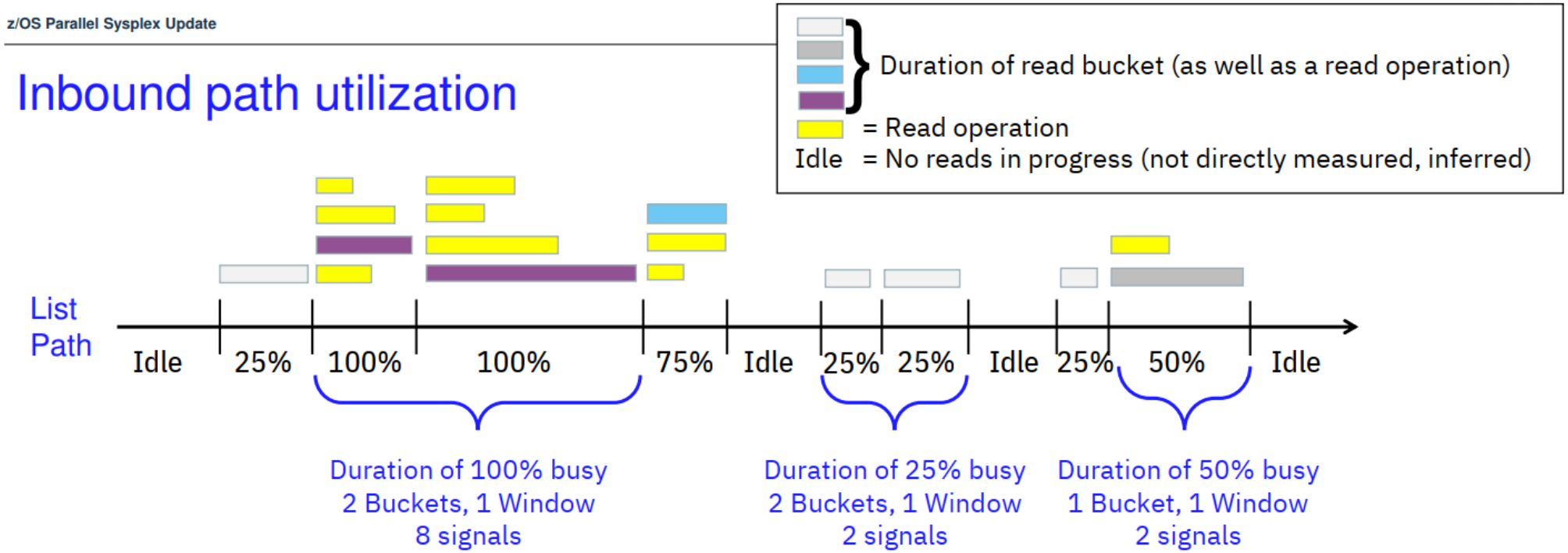
“Buckets” and “Windows”

- XCF also measures total time spent in a path utilization “window”
- Assuming an idle path, a window begins when a bucket of N reads is started
- That window continues as long as another N reads are initiated at the end of the current bucket
- A window ends when the value of N changes
  - If N=0, the path has become idle
  - A different N>0 starts a new window at the new utilization

Buckets and Windows – Illustrated (IBM)

z/OS Parallel Sysplex Update

Inbound path utilization



Utilization	#Buckets	#Windows	#Signals	Window Duration
25%	4	3	4	Σ = + ( ) +
50%	1	1	2	Σ =
75%	1	1	3	Σ =
100%	2	1	8	Σ = ( )

RMF Path Usage Statistics Block (x4)

Path Usage Statistics block This block exists four times. The metrics of one array entry are related to the path utilization percentage that is indicated in metric R742PUSG_Percent. The path usage statistics are available for inbound paths only.					
112	70	R742PUSG_TimeSum	8	binary	Time (in microseconds) this path was in use at the indicated percent utilization.
120	78	R742PUSG_TimeSsq	8	binary	Squared microseconds this path was in use at the indicated percent utilization.
128	80	R742PUSG_Time#	4	binary	Number of times this path was in use at the indicated percent utilization.
132	84	R742PUSG_SigCnt	4	binary	Number of signals sent for this usage entry.
136	88	R742PUSG_Percent	4	binary	Percent utilization that this entry represents.
140	8C	*	4	*	Reserved.
End of four instances of Path Usage Statistics block					

Calculated Timing Metrics

- Path utilization = sum(TimeSum fields for all 4 blocks) / interval
- For each of the 4 utilizations (25/50/75/100)
  - Time per bucket = TimeSum / (SigCnt / (Percent / 25))
    - Percent / 25 = number of active reads for that utilization
  - Time per window = TimeSum / Time#
  - Time per read = TimeSum / SigCnt

Analyzing New Path Usage Metrics

Sample RMF Report (IBM)

Number of signals received

Duration of usage at indicated utilization (milliseconds)

Number of signals received at indicated utilization

New

TOTAL SAMPLES = 899 XCF PATH STATISTICS

INBOUND TO SYS2

FROM SYSTEM	T Y P	FROM/TO DEVICE, OR STRUCTURE	REQ IN	BUFFERS UNAVAIL	TRANSFER TIME	NO BUF TIME	NO BUF	USAGE UTIL%	IN USE TIME	IN USE	SIGNALS
SYS1	S	IXCPATH02	34,685	26	0.100	1.843	28	25	487.896	9,746	14,609
								50	173.741	2,372	7,144
								75	101.694	1,845	5,700
								100	149.782	1,155	7,232
	S	IXCPATH03	62,408			[illegible]93	63	25	819.731	16,712	23,056
								50	305.195	4,318	12,972
								75	206.199	3,816	11,892
								100	275.711	2,459	14,488

Utilization%
25% = 1 of 4
50% = 2 of 4
75% = 3 of 4
100% = 4 of 4
buffers in use

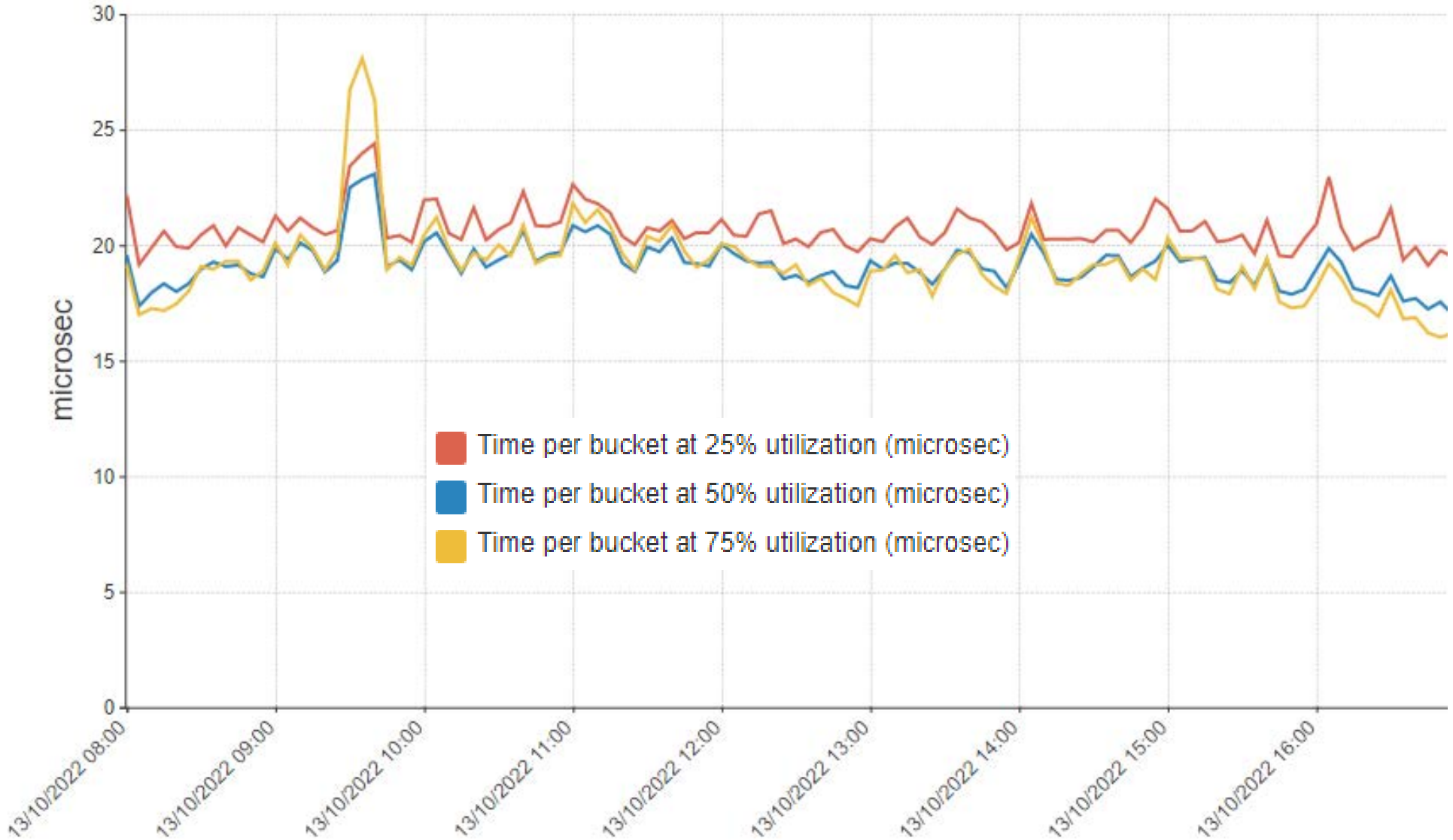
Util%	#bkts	#wndw	average mics per Bkt	Wndw	Read
25	14609	9746	33	50	33
50	3572	2372	49	73	24
75	1900	1845	54	55	18
100	1808	1155	83	130	21

Number of "windows"

15-minute interval = 900,000 milliseconds
IXCPATH02 was utilized for 913 of them
IXCPATH03 was utilized for 1607 of them
(sum in use time)

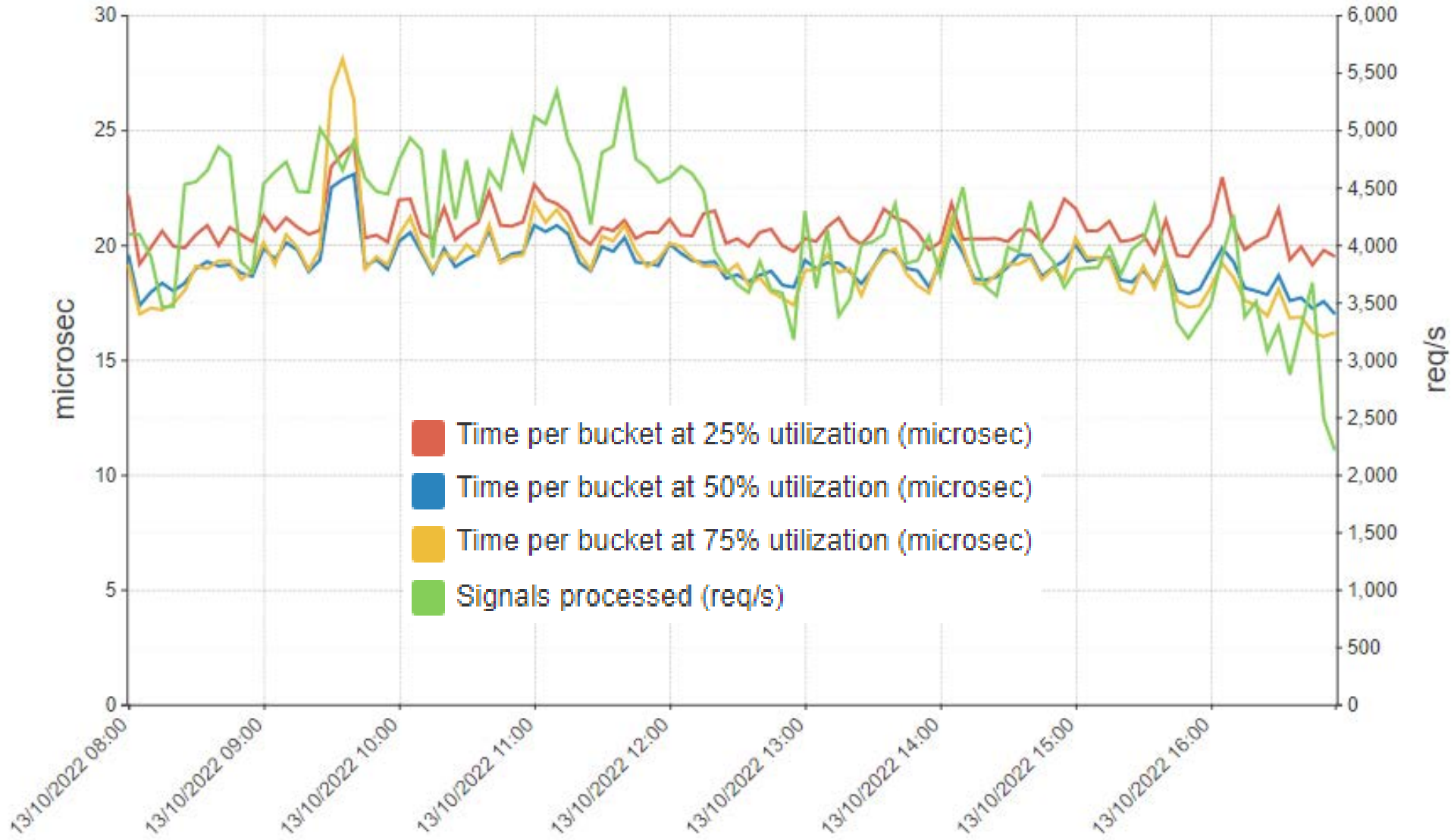
Time per Bucket by Utilization

For System ID ‘SYS1’ with System ID ‘SYS2’



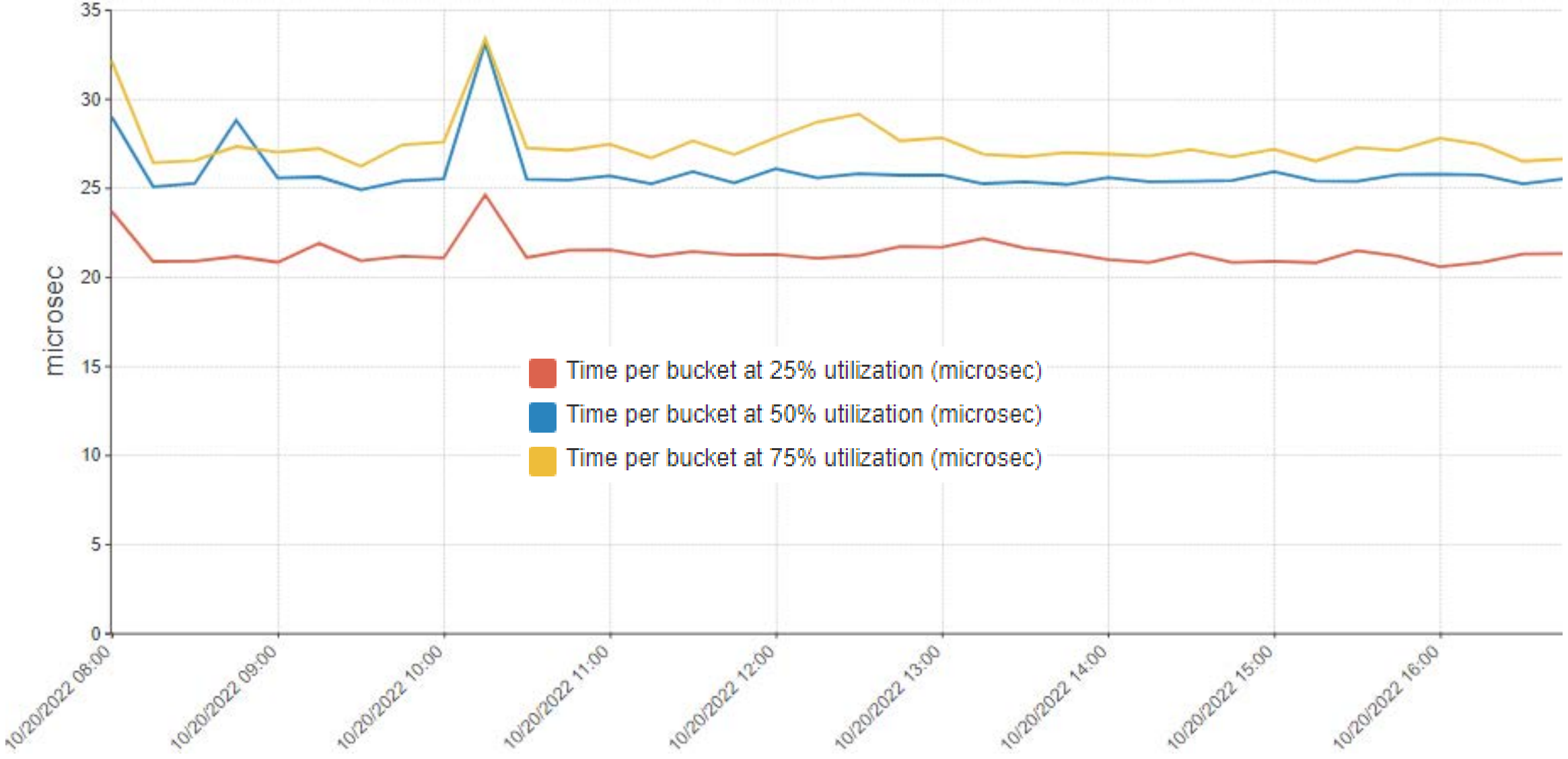
Time per Bucket by Utilization with Inbound Signal Volume

For System ID ‘SYS1’ with System ID ‘SYS2’



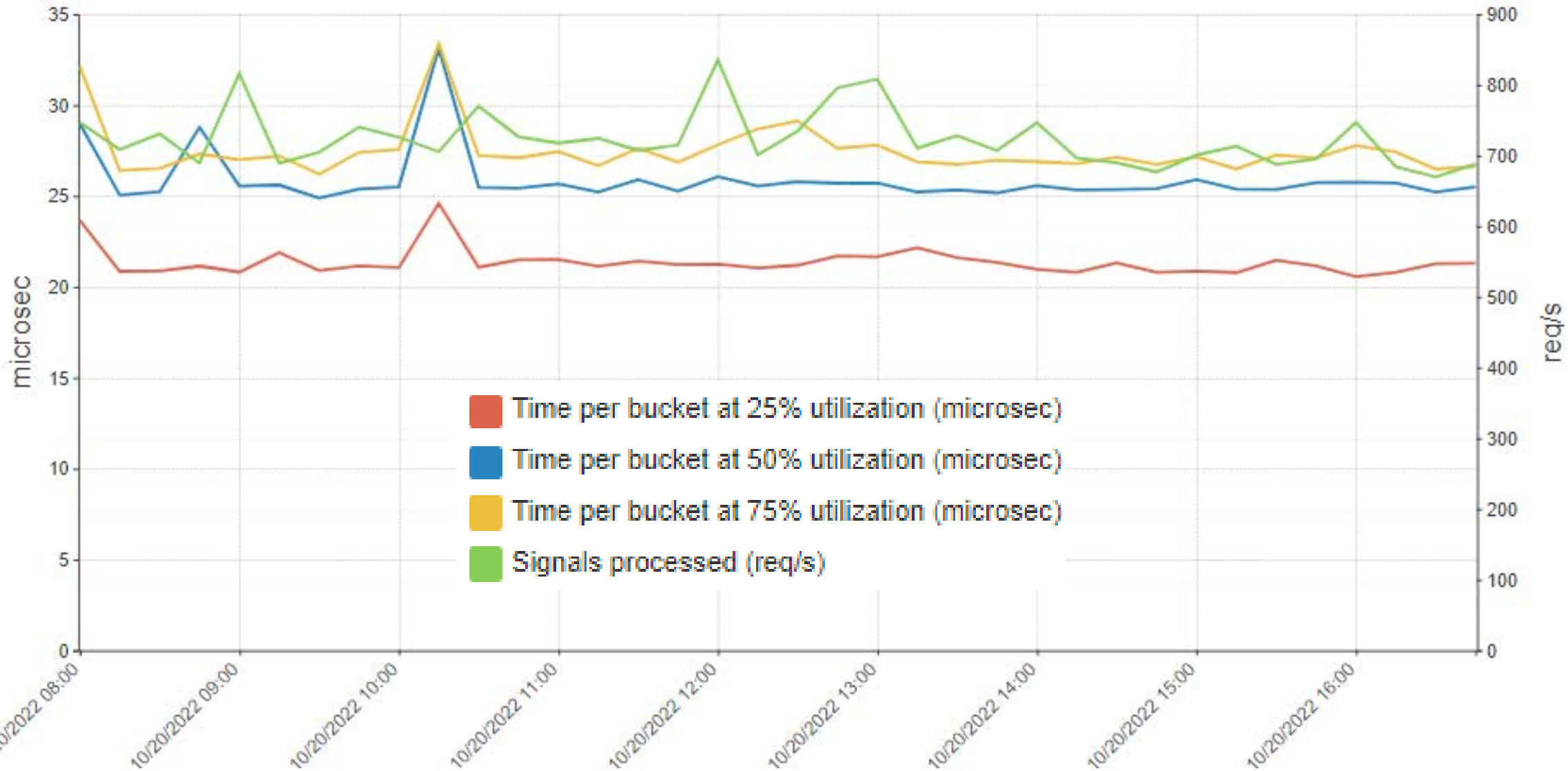
Time per Bucket by Utilization

For System ID ‘SYS3’ with System ID ‘SYS4’



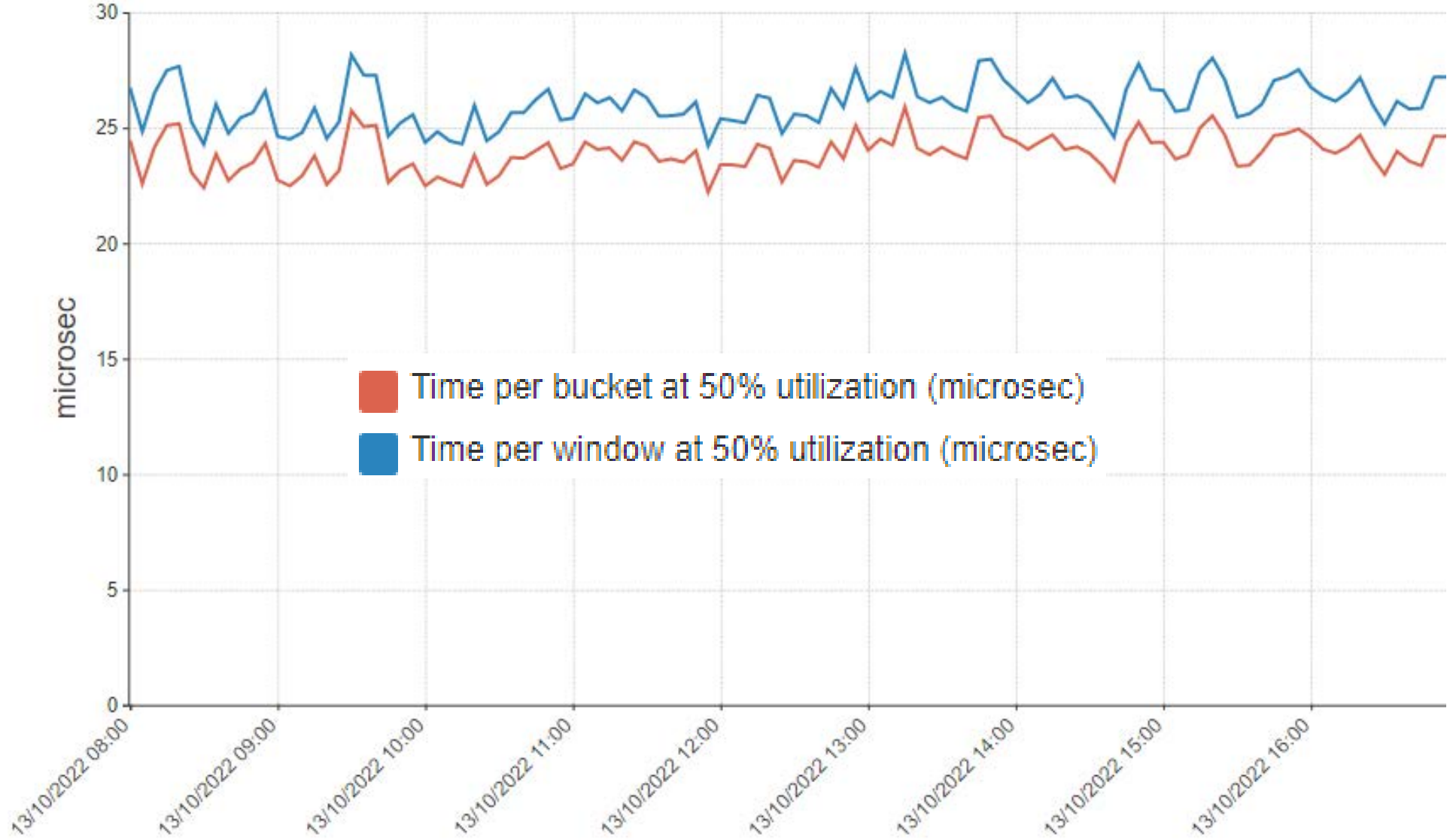
Time per Bucket by Utilization with Inbound Signal Volume

For System ID ‘SYS3’ with System ID ‘SYS4’



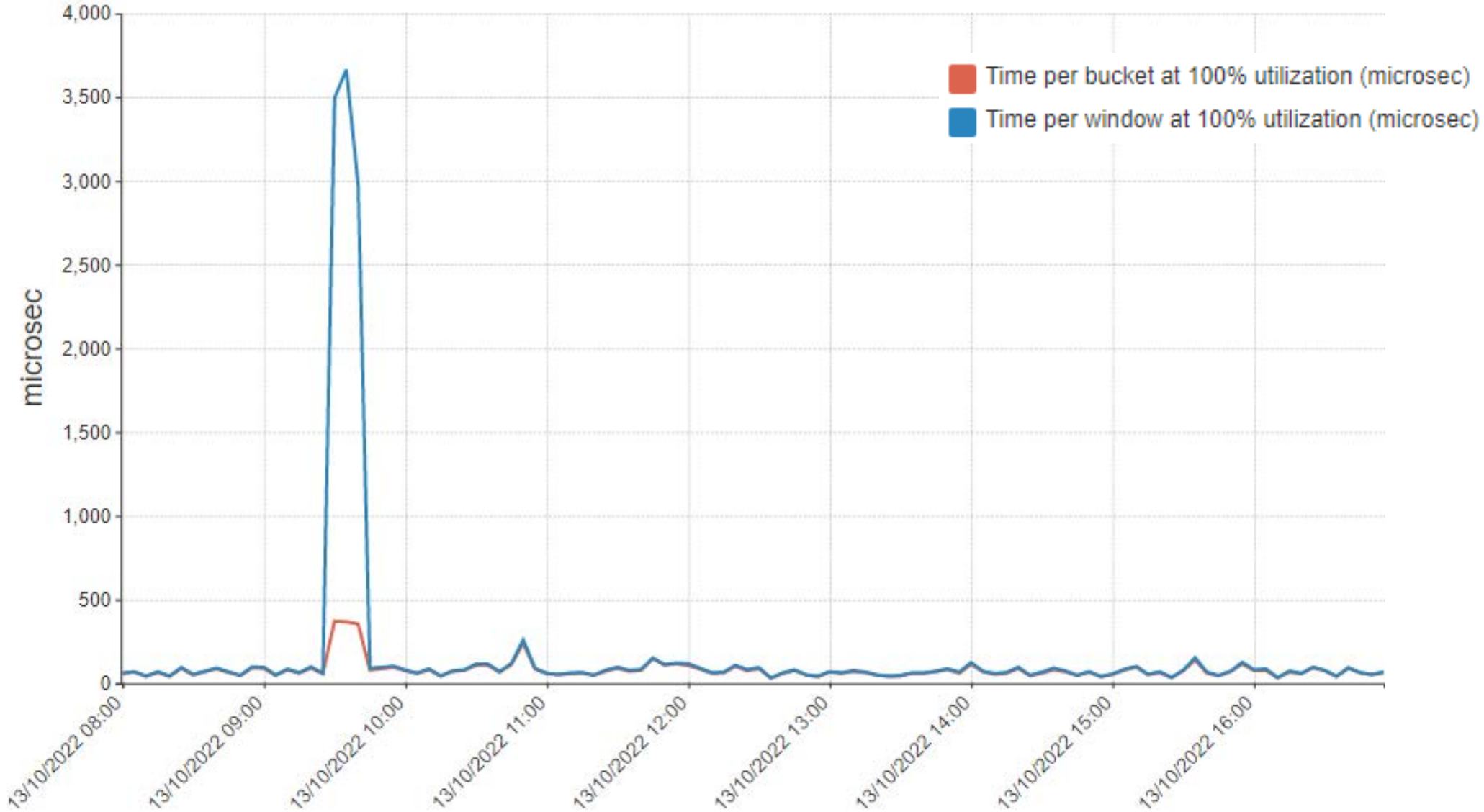
Timing Metrics at 50% Utilization

For System ID ‘SYS1’ with System ID ‘SYS2’



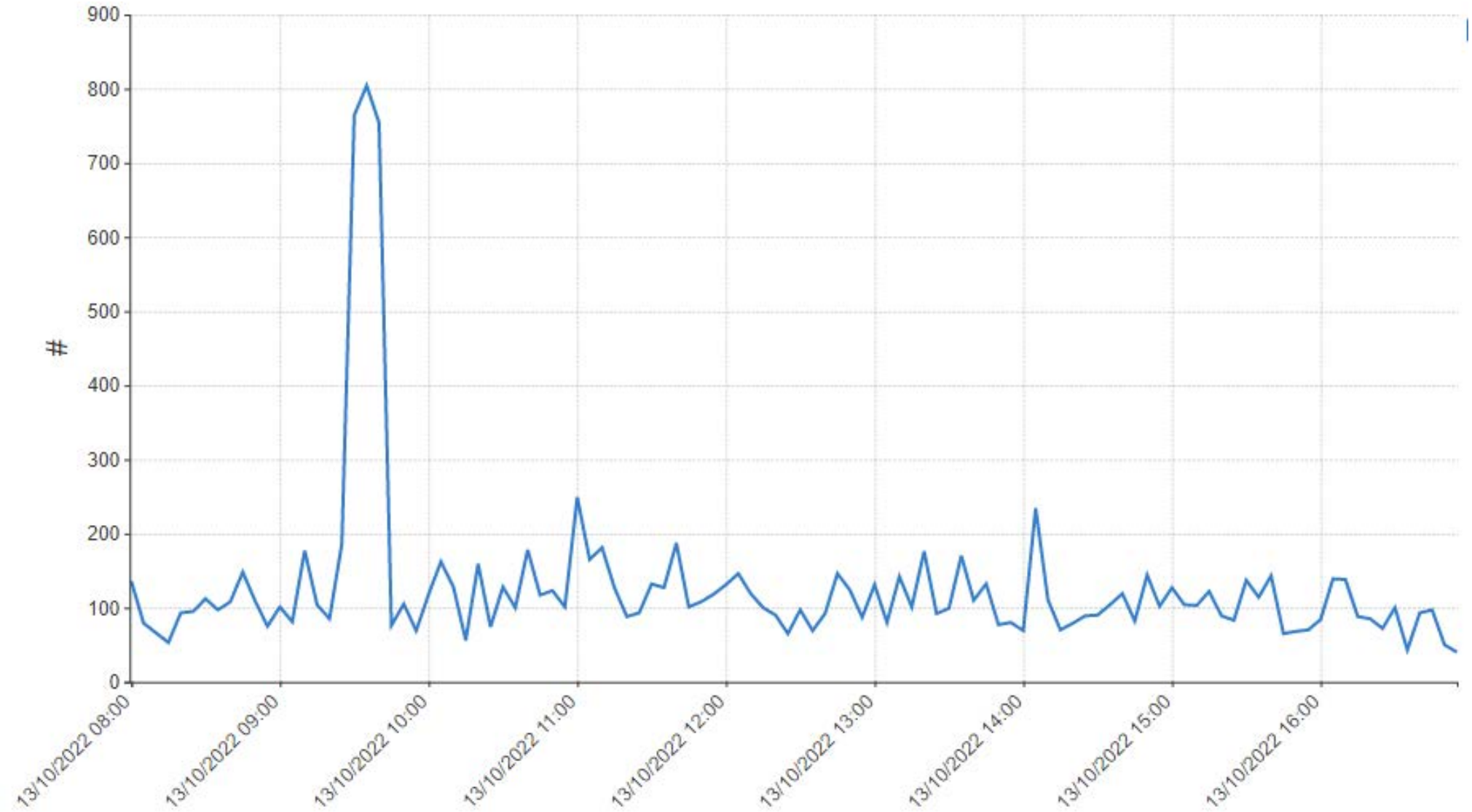
Timing Metrics at 100% Utilization

For System ID ‘SYS1’ with System ID ‘SYS2’



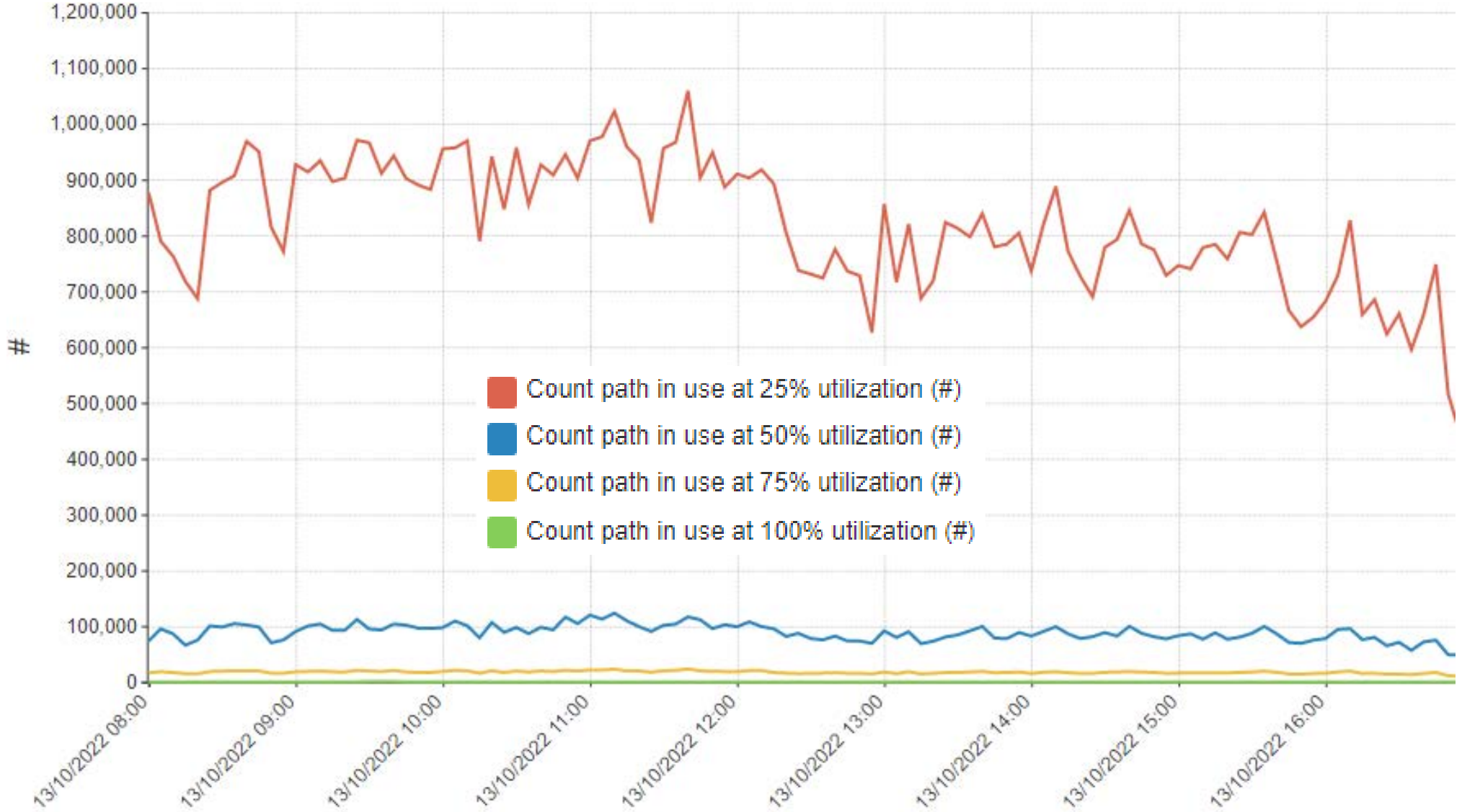
Count Path in Use at 100% Utilization

For System ID ‘SYS1’ with System ID ‘SYS2’



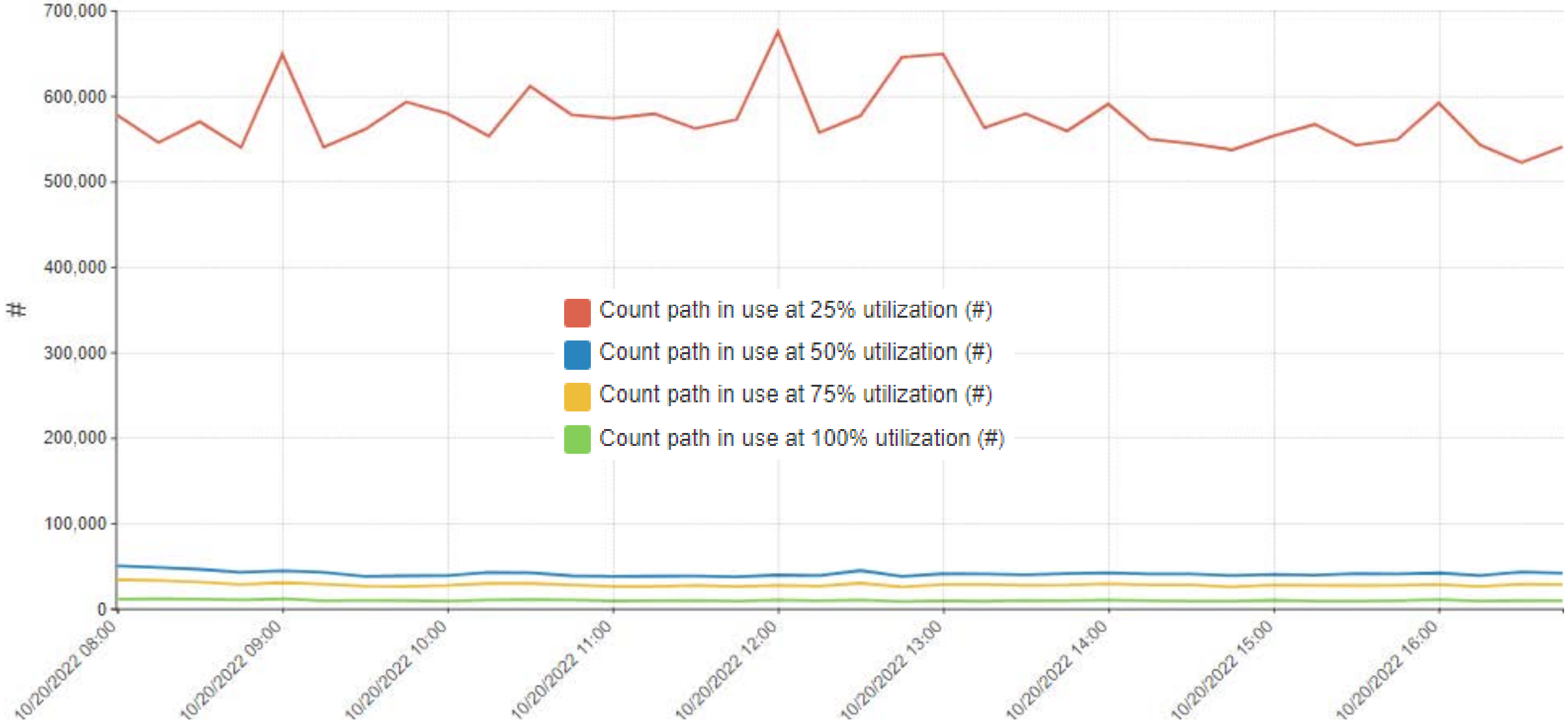
Counts Path in Use by Utilization

For System ID ‘SYS1’ with System ID ‘SYS2’



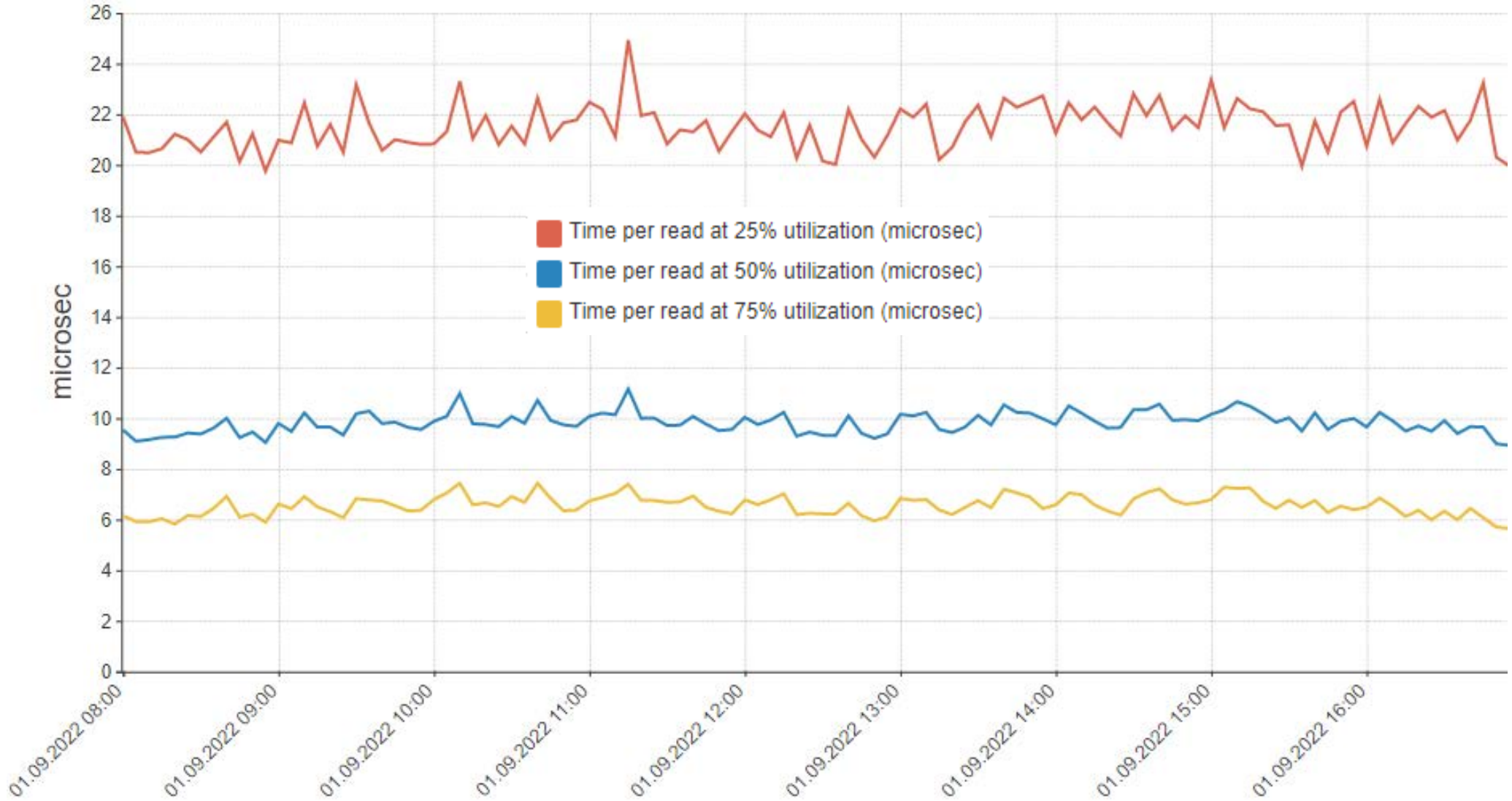
Counts Path in Use by Utilization

For System ID ‘SYS3’ with System ID ‘SYS4’



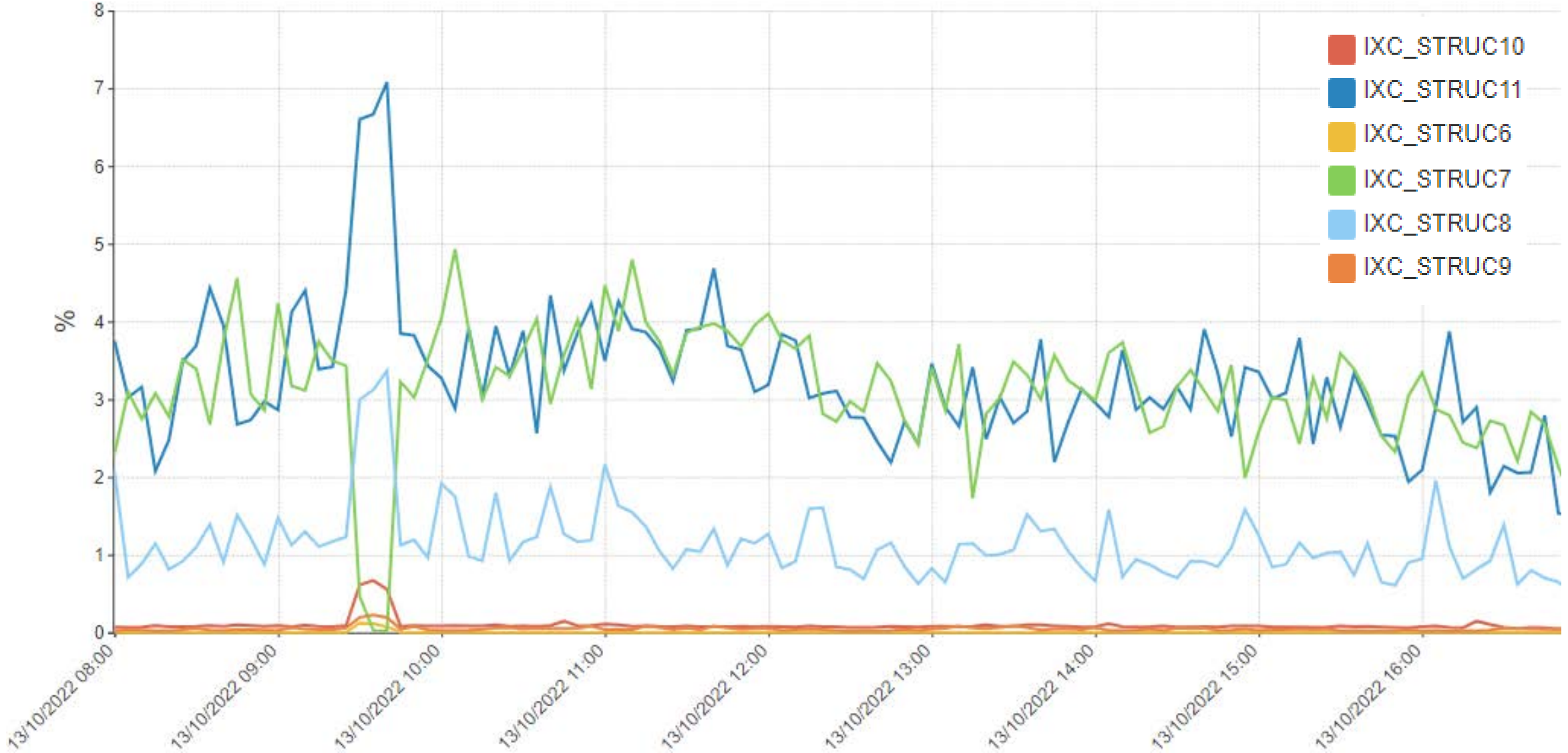
Time per Read by Utilization

For System ID ‘SYS1’ with Target System ID ‘SYS2’



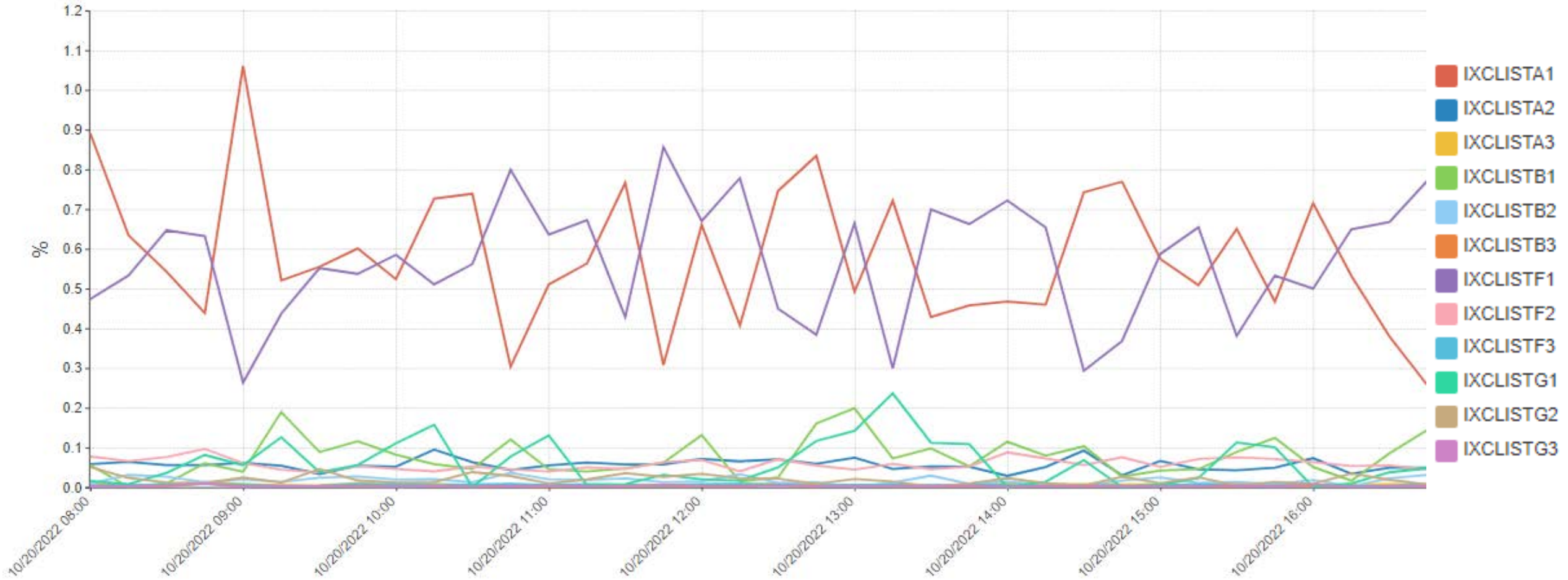
Path Utilization

For System ID ‘SYS1’ with System ID ‘SYS2’ by CF Structure Name



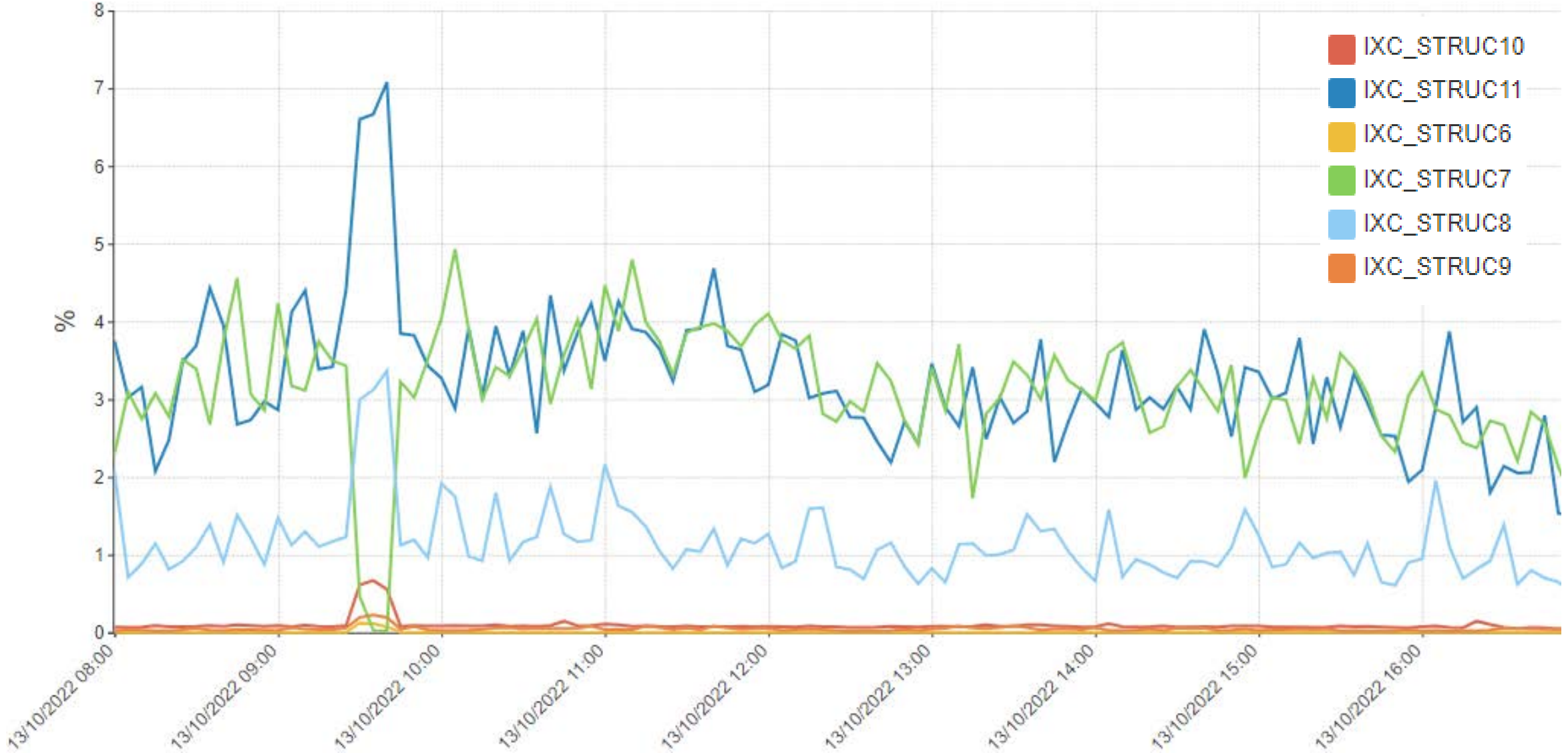
Path Utilization

For System ID ‘SYS3’ with System ID ‘SYS4’ by CF Structure Name



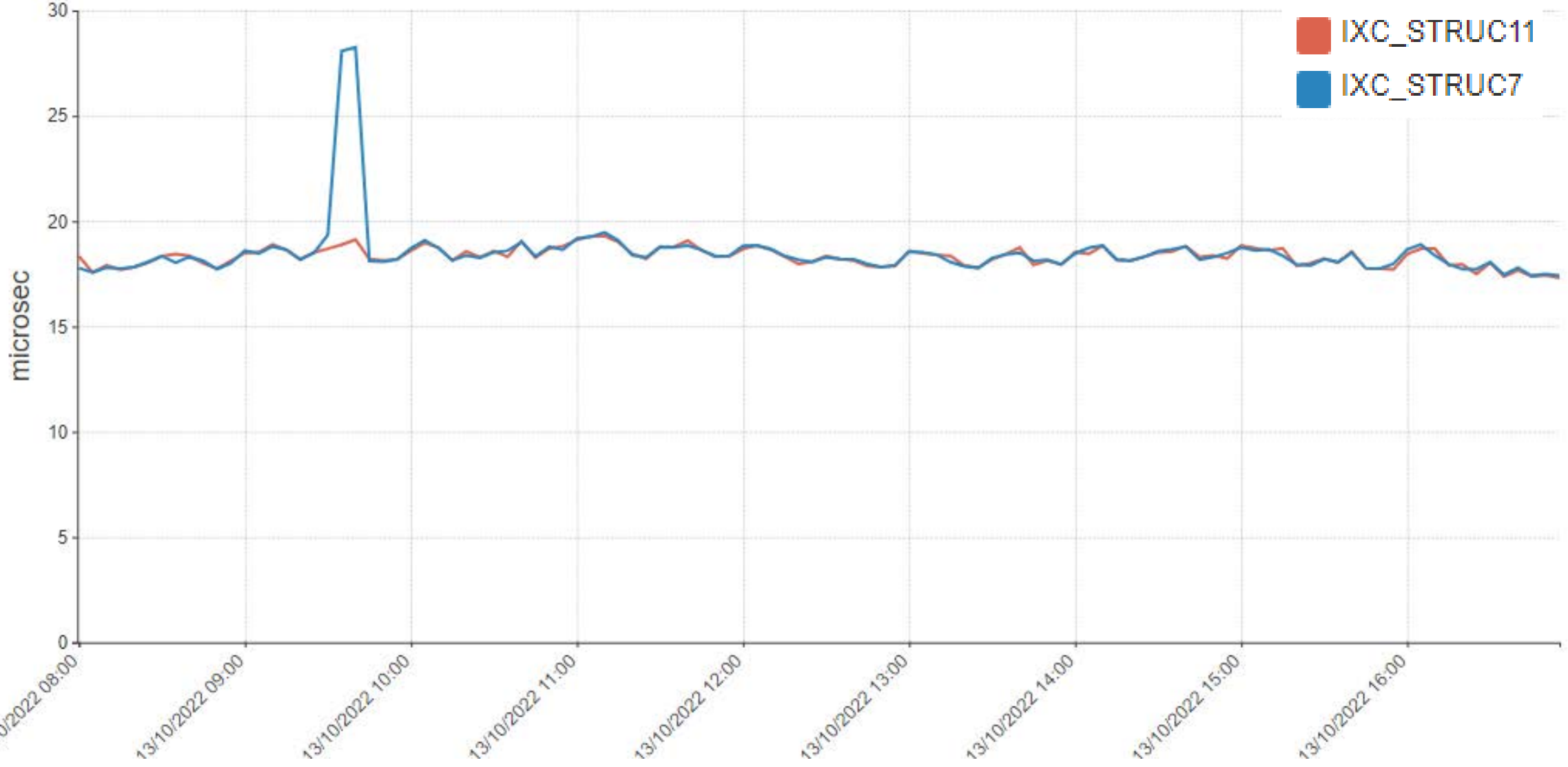
Path Utilization

For System ID ‘SYS1’ with System ID ‘SYS2’ by CF Structure Name



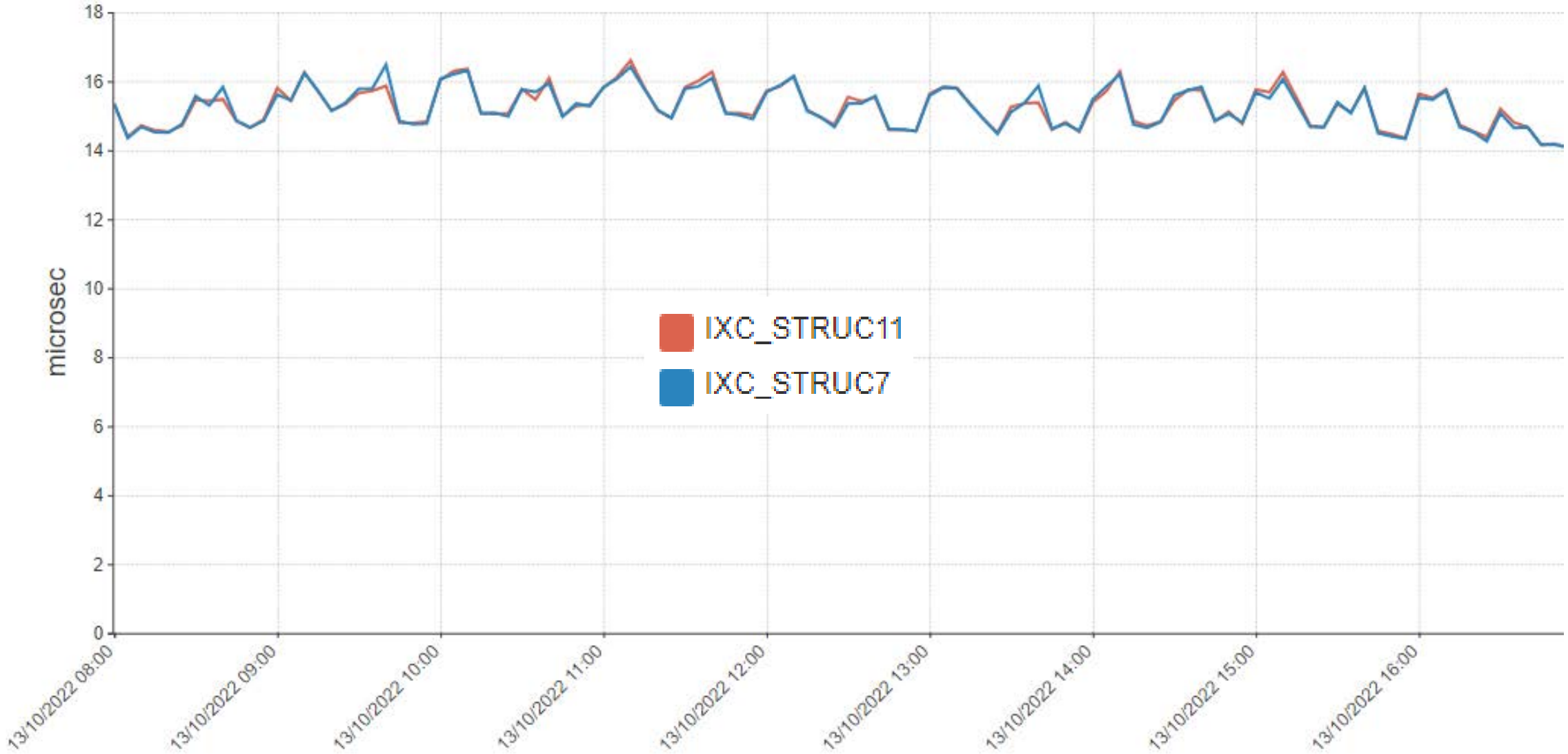
Time per Bucket at 25% Utilization

For System ID ‘SYS1’ with Target System ID ‘SYS2’ by CF Structure Name



Coupling Facility Asynchronous Service Time

For System ID ‘SYS1’ by CF Structure Name

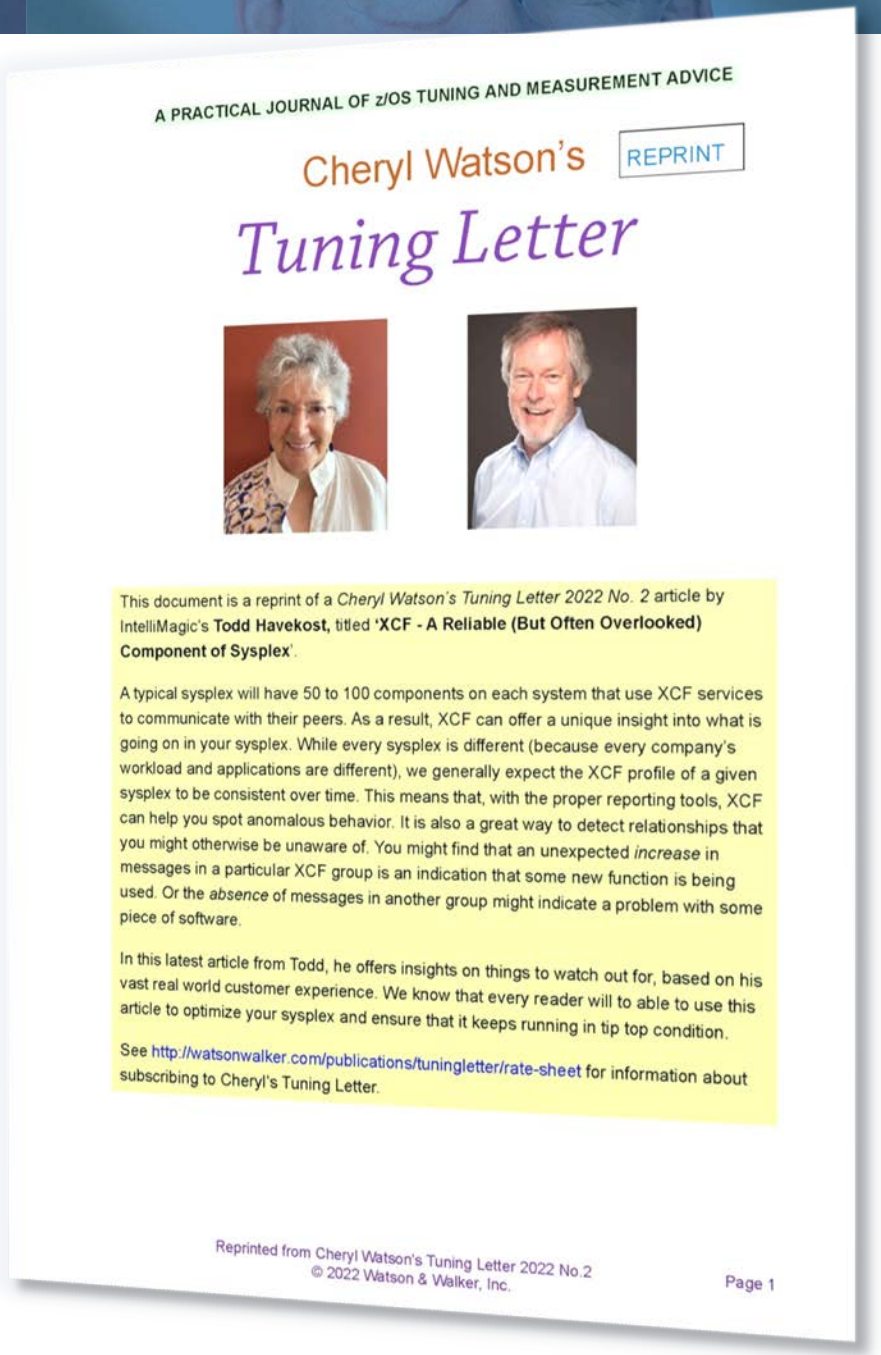


Resources

Cheryl Watson's Tuning Letter

- XCF series of articles
  - Transport Class Simplification – Tuning Letter 2020 No. 2
  - XCF: A Reliable (But Often Overlooked) Component of Sysplex – TL 2022 No. 2
  - Using New XCF Metrics to Optimize XCF Buffer Use (Part 1) – TL 2022 No. 3
  - [Article 4 in XCF series] – TL 2022 No. 4

- To receive reprints of IntelliMagic written articles in the Tuning Letter contact: sales@intellimagic.com



A PRACTICAL JOURNAL OF z/OS TUNING AND MEASUREMENT ADVICE

Cheryl Watson's REPRINT

Tuning Letter

This document is a reprint of a *Cheryl Watson's Tuning Letter 2022 No. 2* article by IntelliMagic's **Todd Havekost,** titled **'XCF - A Reliable (But Often Overlooked) Component of Sysplex'**.

A typical sysplex will have 50 to 100 components on each system that use XCF services to communicate with their peers. As a result, XCF can offer a unique insight into what is going on in your sysplex. While every sysplex is different (because every company's workload and applications are different), we generally expect the XCF profile of a given sysplex to be consistent over time. This means that, with the proper reporting tools, XCF can help you spot anomalous behavior. It is also a great way to detect relationships that you might otherwise be unaware of. You might find that an unexpected *increase* in messages in a particular XCF group is an indication that some new function is being used. Or the *absence* of messages in another group might indicate a problem with some piece of software.

In this latest article from Todd, he offers insights on things to watch out for, based on his vast real world customer experience. We know that every reader will to able to use this article to optimize your sysplex and ensure that it keeps running in tip top condition.

See http://watsonwalker.com/publications/tuningletter/rate-sheet for information about subscribing to Cheryl's Tuning Letter.

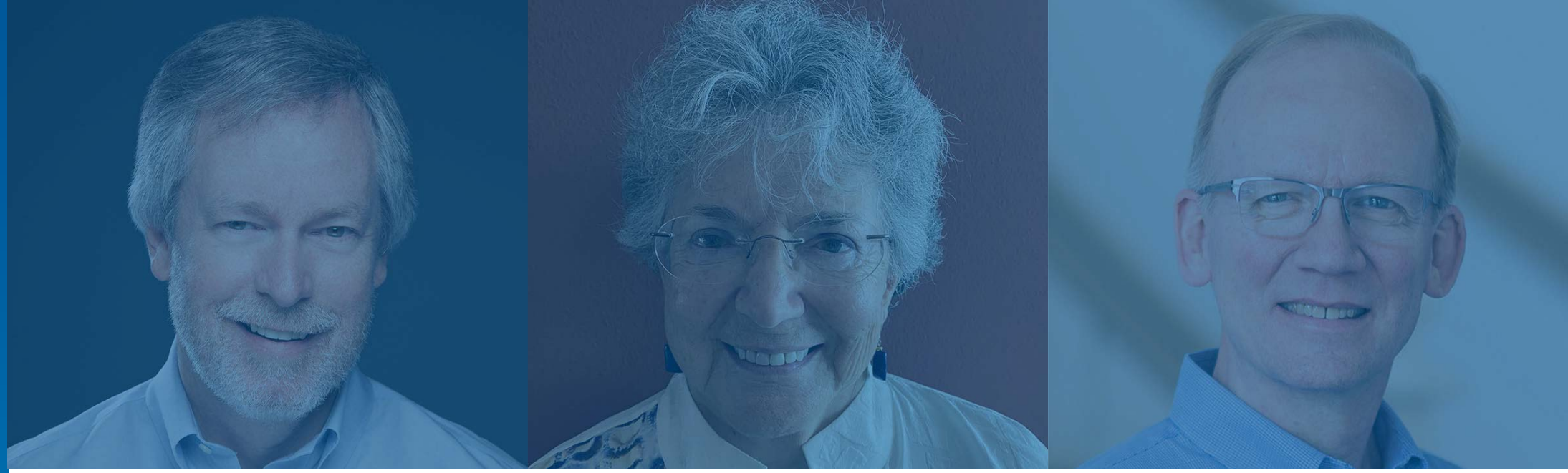
Reprinted from Cheryl Watson's Tuning Letter 2022 No.2
© 2022 Watson & Walker, Inc.

Page 1

Additional Information

- SHARE in Columbus 2022 session *Parallel Sysplex Update*, by **Mark Brooks**.
- z/OS MVS Setting Up a Sysplex, SA23-1399.
- IntelliMagic zAcademy webinar *Insights Iinto New XCF Path Usage Metrics*, by **Todd Havekost** and **Frank Kyne**.

Questions?



Thank you!

zAcademy sessions are available on-demand at
http://www.intellimagic.com/zacademy

For additional information, please contact
sales@intellimagic.com