z/OS 2.4 IBM Education Assistant (IEA)

Solution (Epic) Name: XCF Transport Class Simplification

Element(s)/Component(s): z/OS XCF

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.

Session Objectives

- A little background on XCF Transport Classes to provide some context
- Where are we headed?
- What do you get in z/OS V2R4?
  - How enabled?
  - What changes?
  - Caveats and concerns?

Background

Brief review of Transport Class basics

Background: Transport Classes

- Transport classes segregate message traffic by size, group, or both
- Message traffic is segregated by size for the purpose of:
    - Ensuring timely transfer for small messages
    - Efficient utilization of buffer space
- Message traffic is segregated by group for the purpose of:
    - Isolating an ill-behaved group to protect others from harm
    - Dedicating XCF signal resources to a favored group so that it need not compete with others. Ostensibly, the group will:
        - Never be denied service
        - Never experience transfer delays

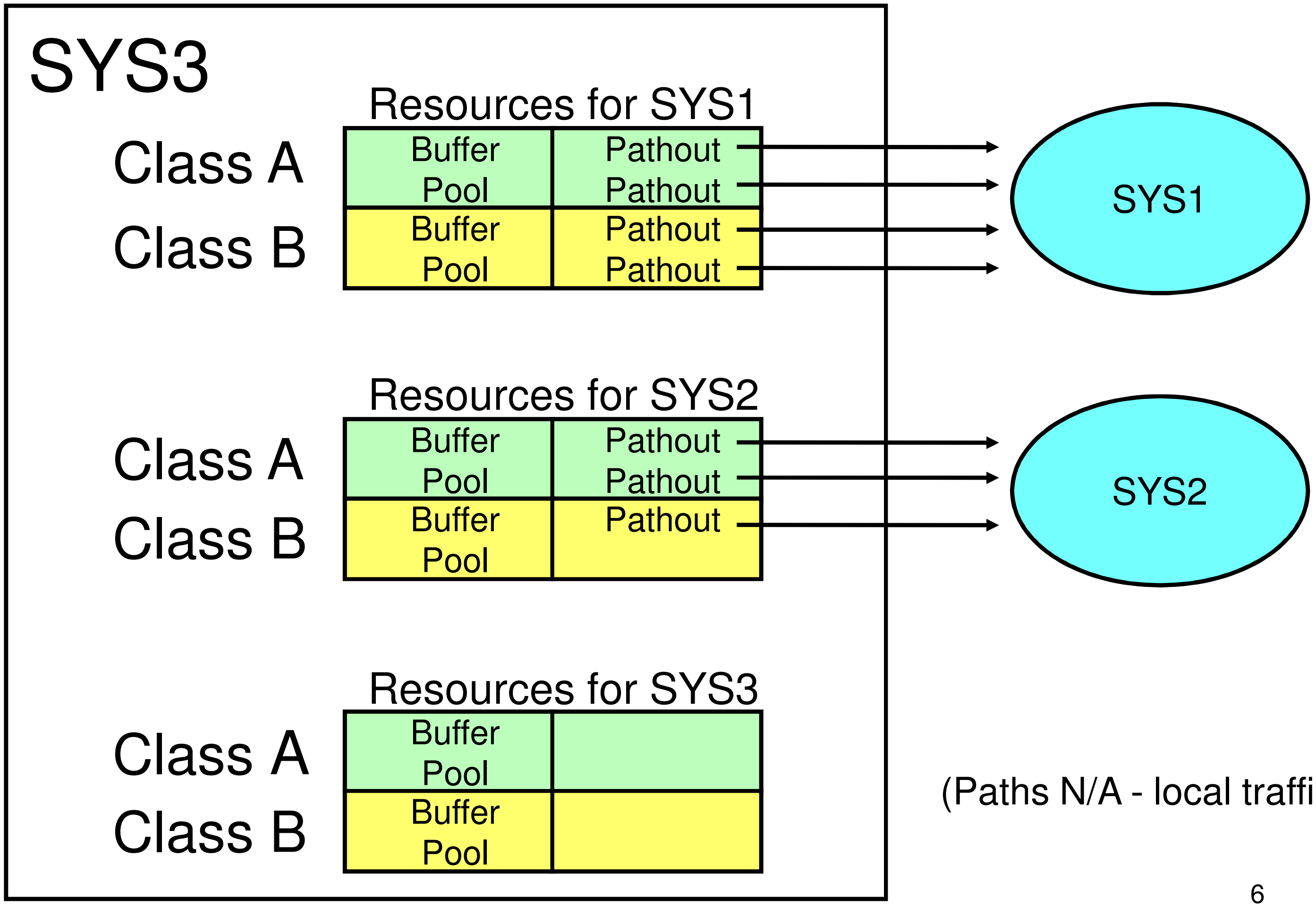
Background: Transport Classes

Transport class has buffers and signal paths (unless local).

Each target system has it’s own dedicated signal resources. Never shared between systems.

The same class definition applies to all target systems.

In practice, the signal traffic patterns are often vary dramatically on a target system basis.



SYS3

Resources for SYS1

Class A	Buffer Pool	Pathout Pathout
Class B	Buffer Pool	Pathout Pathout

Resources for SYS2

Class A	Buffer Pool	Pathout Pathout
Class B	Buffer Pool	Pathout

Resources for SYS3

Class A	Buffer Pool	
Class B	Buffer Pool	

SYS1

SYS2

(Paths N/A - local traffic)

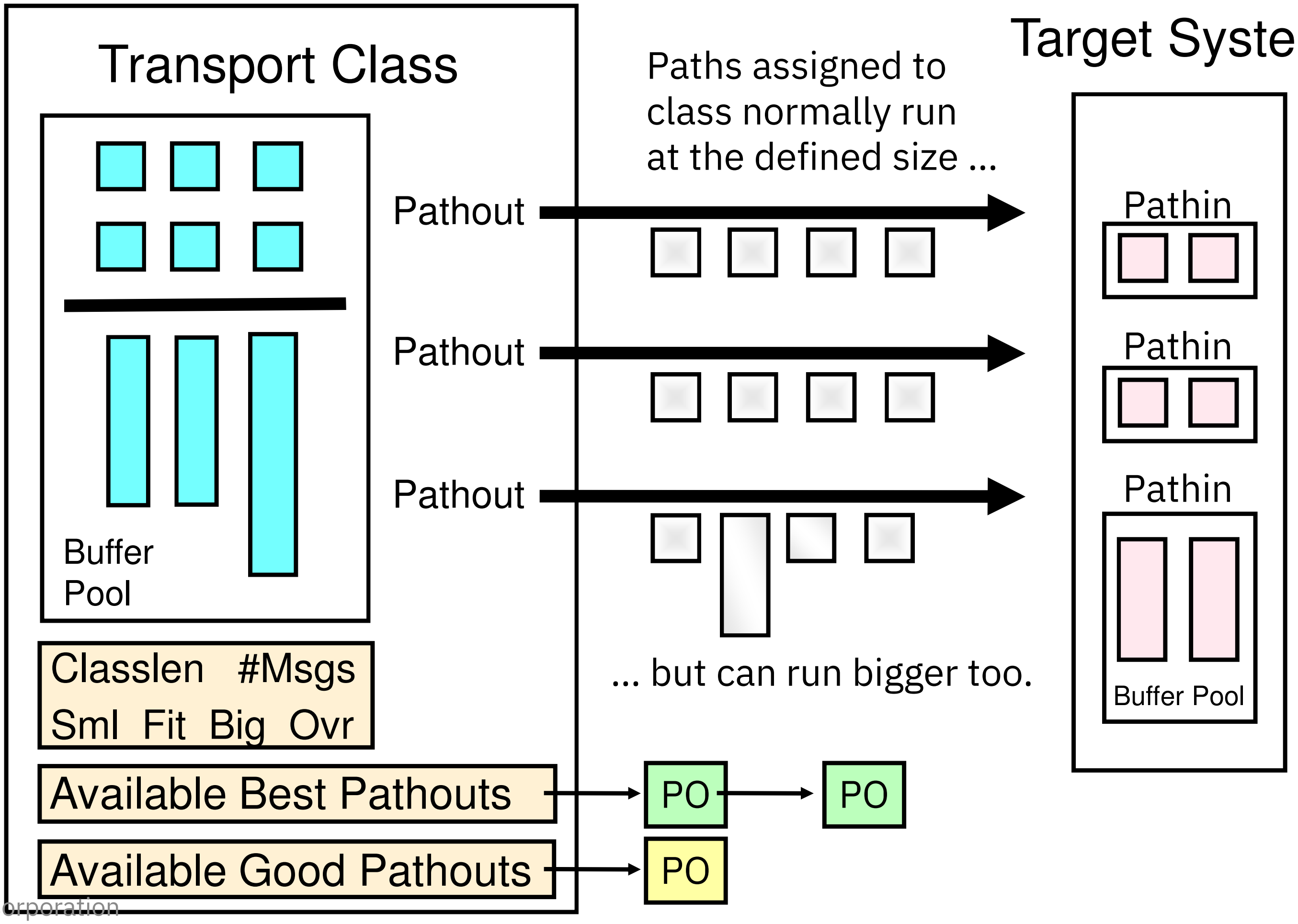
Background: Transport Class

Buffers for signals no bigger than the “defined” size.

Buffers for signals longer than defined size.

Buffer Accounting

Path Accounting (Avail vs Busy)



Transport Class

Buffer Pool

Classlen #Msgs
Sml Fit Big Ovr

Available Best Pathouts

Available Good Pathouts

Pathout

Pathout

Pathout

Paths assigned to class normally run at the defined size ...

... but can run bigger too.

PO

PO

PO

Target System

Pathin

Pathin

Pathin

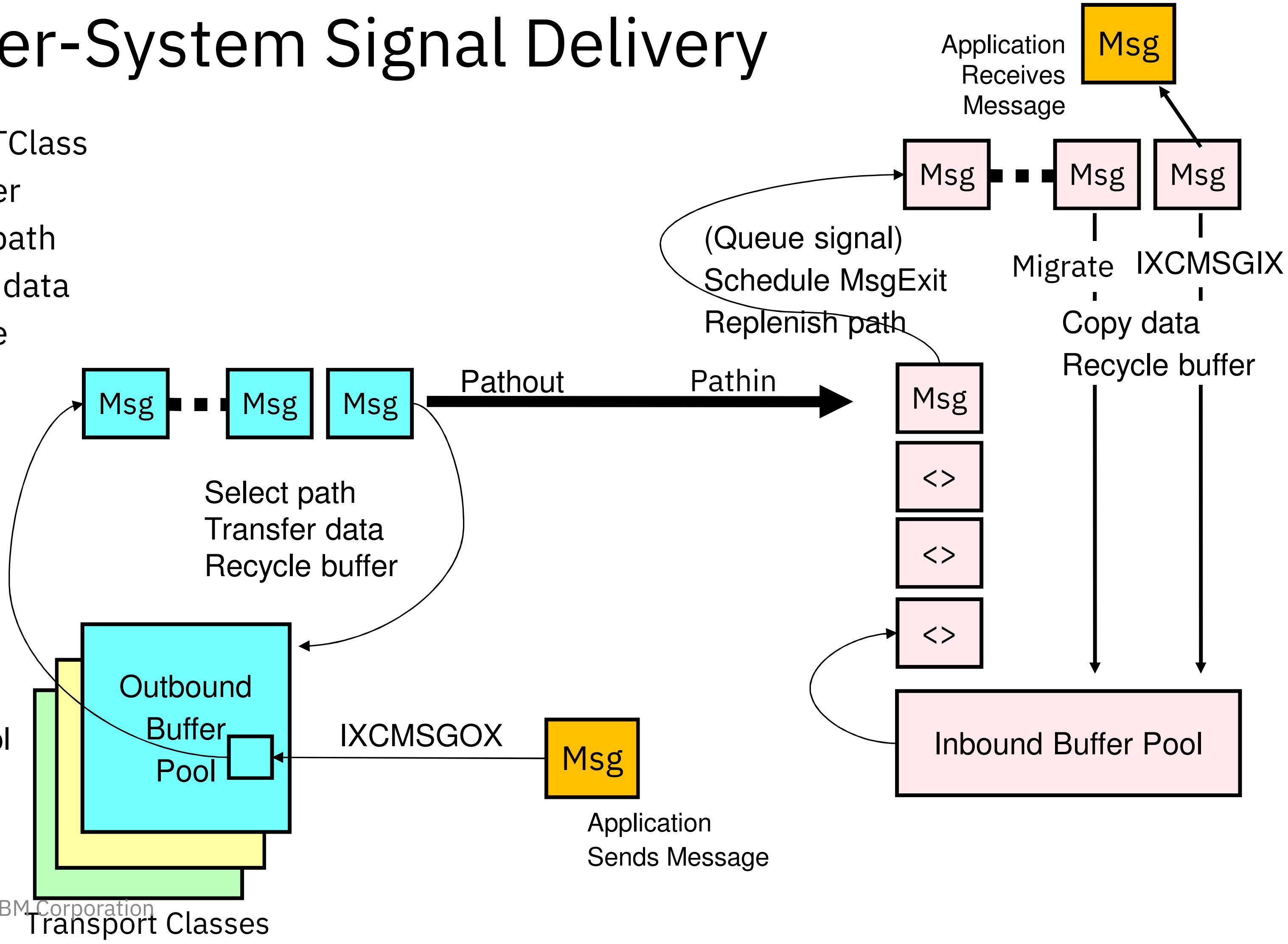
Buffer Pool

Inbound paths must be ready to receive whatever size signal is demanded by the outbound side.

Changes must be renegotiated through a process that sends additional control signals and temporarily quiesces signal transfer.

Inter-System Signal Delivery

- Choose TClass
- Get buffer
- Choose path
- Transfer data
- Schedule SRB
- Call user msg exit



Select pool
Get buffer
Copy data

Where are we headed?

The installation can achieve the high level of resiliency, availability, and performance expected of sysplex without having to plan, define, monitor, or tune z/OS XCF transport classes.

Why eliminate the need to define Transport Classes?

- Simplification
  - Remove this burden from customers and all the IBM personnel that have to analyze, explain, make recommendations, etc.
  - Hard to provide clear guidance as to how classes should be configured; more art than science
- Avoid Outages
  - Transport Classes often not well understood, mistakes are made that lessen the resiliency of the sysplex and so permit avoidable outages
  - A Transport Class can help isolate an ill behaved member, but not really practical since you have to know in advance who is going to cause trouble
    - Such as delays and overwhelming bursts
- Self-optimization
  - Static definitions are not well suited to the dynamics of the sysplex
    - Can lead to inefficient use of resources (idle paths, excess storage)
  - System should automatically apply resources where needed most
    - For example, to handle a burst of messages or a change in signal patterns

To eliminate the need to define Transport Classes

XCF must automatically handle the problems that transport classes were intended to address:

- Timely message transfer
  - Maintain signal throughput and minimize signal delivery times, especially for the small messages that are typically most predominant in the sysplex
  - Provides better resiliency (minimize delays and queueing)
- Efficient utilization of signal resources
  - Helps minimize cost
  - Provides better resiliency (more capacity)

Size segregation

Group segregation

- Isolation of ill behaved members
  - Avoid sympathy sickness so that problems with signal delivery for one member don’t negatively impact signal delivery for other members
- Fair access to signal resources
  - Don’t allow one member to monopolize the signal resources to the detriment of signal delivery for other members

To eliminate the need to define Transport Classes

XCF must automatically handle the problems that transport classes were intended to address:

- Timely message transfer
  - Maintain signal throughput and minimize signal delivery times, especially for the small messages that are typically most predominant in the sysplex
  - Provides better resiliency

urce

oble

ery f

lize

signal delivery for other members

In the early days of sysplex, there was a significant difference in transfer time for small vs large signals. Today, this difference is much, much smaller and signal throughput is much greater. The relative delay arising from a small signal following a large one vs that of following a small signal is negligible. Intermixing signal sizes on a signal path will not have any discernable impact on the performance of the workload.

For z/OS V2R4, timely message transfer has been a primary focus. Our rationale:

- There will likely be some consolidation of signal paths as transport classes are eliminated, we want to make sure the survivors can sustain the signal load.
- What if there is a workload for which those marginal differences in transfer times for small vs large signals does matter? The ability to sustain higher signal rates will likely mask any impacts that might arise.
- And ....

To eliminate the need to define Transport Classes

XCF must automatically handle the problems that transport classes were intended to address:

- Timely message transfer
  - Maintain signal throughput and minimize si small messages that are typically most pre
  - Provides better resiliency
- Efficient utilization of signal resources
  - Helps minimize cost
  - Provides better resiliency
- Isolation of ill behaved members
  - Avoid sympathy sickness so that problems don’t negatively impact signal delivery for c
- Fair access to signal resources
  - Don’t allow one member to monopolize the signal delivery for other members

These days, the more compelling reason for segregating signals by size has been to ensure that we can **maximize the number of signal buffers** available to the inbound path within the constraint of its MAXMSG limit. In general, more buffers tends to improve resiliency and helps maintain throughput.

Since our solution for eliminating the need to segregate signals by size uses the maximum size inbound buffer, we effectively **minimize the number of signal buffers** available to the inbound path. Even more impetus to focus on timely transfer and signal throughput!

To eliminate the need to define Transport Classes

XCF must automatically handle the problems that transport classes were intended to address:

- Timely message transfer
  - Maintain signal throughput and minimize signal delivery times, especially for the small messages that are typically most pre
  - Provides better resiliency
- Efficient utilization of signal resources
  - Helps minimize cost
  - Provides better resiliency
- Isolation of ill behaved members
  - Avoid sympathy sickness so that problem don’t negatively impact signal delivery fo
- Fair access to signal resources
  - Don’t allow one member to monopolize t signal delivery for other members

“Message Isolation” was delivered in z/OS V2R2. It is a foundational, incremental step along the journey towards the goal of eliminating the need to define transport classes for the purpose of segregating signals by group. Might provide “good enough” isolation, but it is not sufficient to make the claim that there is no need to define transport classes to segregate signals by group.

To eliminate the need to define Transport Classes

XCF must automatically handle the problems that transport classes were intended to address:

- Timely message transfer
  - Maintain signal throughput and minimize signal delivery times, especially for the small messages that are typically most pr
  - Provides better resiliency
- Efficient utilization of signal resources
  - Helps minimize cost
  - Provides better resiliency
- Isolation of ill behaved members
  - Avoid sympathy sickness so that problem don't negatively impact signal delivery fo
- Fair access to signal resources
  - Don't allow one member to monopolize th signal delivery for other members

In general, we don't see field problems that would have been avoided if only we had had fair access to the XCF signal service. What I think of as "isolation" issues are more prevalent and problematic.
But until we can guarantee fairness, we can't claim there is no need to define transport classes to segregate signals by group.

So what do you get?

Elimination of the need for transport classes to segregate signals by size.

(Elimination of need for group segregation is not included with this solution).

Overview

- Who (Audience)
  - System programmers, sysplex architects, performance analysts, capacity analysts, diagnosticians
- What (Solution)
  - XCF will internally manage the signal resources to provide timely delivery of signals in a sysplex independent of any Transport Class definitions created purely for size segregation.
- Wow (Benefit / Value, Need Addressed)
  - Simplification: You no longer need to define, monitor, tune, or manage XCF Transport Class definitions to segregate signals purely by size.
  - Simplification: You need only configure an appropriate number of signal paths.
  - Resilience: Less potential for non-optimal transport class definitions to negatively impact signal delivery. Probably more signal capacity.[1]

[1] Given the same configuration.

The solution

- XCF will intermix signal sizes on “XCF Managed” signal paths as it sees fit while maintaining signal throughput and timely signal transfer, especially for small signals
  - Thus eliminating the need for transport class based signal size segregation
  - Any available path can be used for any size signal
    - Could do this today if we like. The challenge is to achieve this without incurring any discernable impact to signal throughput, system overhead, resiliency, …

We assume adequate signal capacity and system performance.

If inadequate capacity, or systems not performing well:

- Transport classes are irrelevant; they cannot address such issues
- Segregating signal traffic on the sending system does not resolve issues related to the target system being unresponsive

Feedback during development

- On the use of existing transport class definitions:
  - Either honor them as given, or ignore them outright
  - Want to control whether system uses new or old behavior
- Should be able to observe the new behavior
  - Want to be able to see which behavior is in play, new or old
- Focus first on eliminating the need for “size segregation”
  - Applicable to most if not all installations
  - For those that don’t do group segregation, the “problem” is solved
- Don’t change meaning of existing measurements (SMF data)
  - Differences between releases makes it hard to interpret the data

So there is a new switch, XTCSIZE

So there is a new XCF defined pseudo transport class, _XCFMGD

Usage & Invocation

- New XCF FUNCTIONS switch XTCSIZE determines what transport class segregation rules are available to XCF
- When XTCSIZE is DISABLED:
  - XCF signal resources will be managed per traditional transport class segregation rules
- When XTCSIZE is ENABLED:
  - XCF has the option to manage signal resources for selected transport classes per new “XCF Managed” segregation rules
  - Transport Classes subject to being XCF Managed per XTCSIZE are those that are defined purely for size segregation

Interactions & Dependencies

- Exploitation requires:
  - A z/OS V2R4 sending system with XTCSIZE switch ENABLED
  - A target system running z/OS V2R4
    - Includes self, but the more interesting case is when target is some other system
- Software Dependencies
  - None.
- Hardware Dependencies
  - None.
- Exploiters
  - Implicit (potentially any user of XCF Signal Service)

Migration & Coexistence Considerations

- You will always see the new pseudo _XCFMGD “transport class”
  - But does not impact system behavior unless XTCSIZE is ENABLED.
- No toleration/coexistence APARs/PTFs are needed.
  - z/OS V2R4 manages signal resources per the traditional transport class segregation rules when communicating with a system running an older z/OS release (the XTCSIZE switch setting is irrelevant).
  - So until all systems in the sysplex are running z/OS V2R4, you will still need your traditional transport class definitions.
- Upgrade actions:
  - XTCSIZE is ENABLED by default. So ...
  - If you don’t want the new behavior, XTCSIZE must be DISABLED:
    - COUPLExx parmlib member: FUNCTIONS DISABLE(XTCSIZE)
    - Operator command: SETXCF FUNCTIONS,DISABLE=XTCSIZE

Installation

- IPL z/OS V2R4
  - XTCSIZE is ENABLED by default.
  - You will get new behavior upon IPL.

- Note that XTCSIZE is a “local” switch.
  - A system obeys its own local switch setting
  - If ENABLED, there is the potential for new behavior when sending signals to any system in the sysplex that is also running z/OS V2R4
  - The XTCSIZE switch setting on some other system in the sysplex does not influence local system behavior at all.
  - Purely a “send side” setting.

Which classes are “XCF Managed” per XTCSIZE?

```
CLASSDEF CLASS(SML) CLASSLEN(956) GROUP(UNDESIG)
CLASSDEF CLASS(MED) CLASSLEN(12000)
CLASSDEF CLASS(BIG) CLASSLEN(32000)

CLASSDEF CLASS(BADGUY0) CLASSLEN(956)  GROUP(MYAPP1)
CLASSDEF CLASS(BADGUY1) CLASSLEN(4028) GROUP(MYAPP1,UNDESIG)
CLASSDEF CLASS(BADGUY2) CLASSLEN(8124) GROUP(UNDESIG)
```

XTCSIZE selects those transport class definitions that do NOT have a group explicitly assigned to the class.
Which classes have explicitly assigned groups? These will NOT be XCF Managed. All others will be.

Classes BADGUY0 and BADGUY1 have an explicit group assignment, namely group MYAPP1.
GROUP(UNDESIG) is NOT an explicit group assignment. UNDESIG refers to all groups NOT explicitly assigned.

Answer: XTCSIZE would manage/select these classes:
DEFAULT – this class is implicitly defined with GROUP(UNDESIG) if you don’t otherwise change it.
SML - UNDESIG is not an explicit group assignment.
MED - no explicit groups assigned, so pure size.
BIG - no explicit groups assigned, so pure size.
BADGUY2 – UNDESIG is not an explicit group assignment.

BADGUY0 and BADGUY1 do suggest a desire for size segregation, but it’s not “pure size”. They are for group segregation.

A subtle point on class selection

```
CLASSDEF CLASS(SML) CLASSLEN(956) GROUP(UNDESIG)
CLASSDEF CLASS(MED) CLASSLEN(12000)
CLASSDEF CLASS(BIG) CLASSLEN(32000)

CLASSDEF CLASS(BADGUY0) CLASSLEN(956)  GROUP(MYAPP1)
CLASSDEF CLASS(BADGUY1) CLASSLEN(4028) GROUP(MYAPP1,UNDESIG)
CLASSDEF CLASS(BADGUY2) CLASSLEN(8124) GROUP(UNDESIG)
```

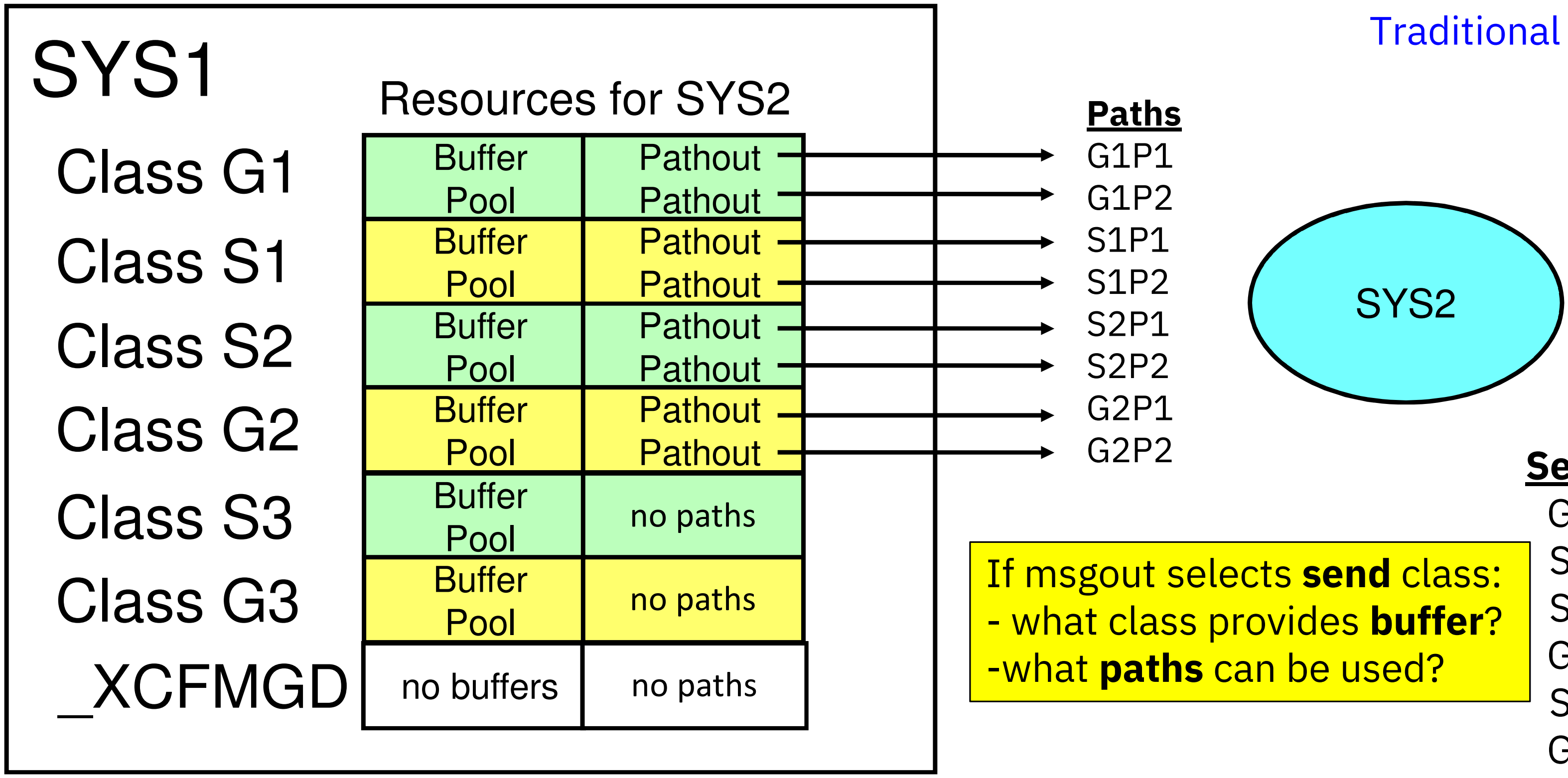
Class BADGUY1 is actually available for use by any group, since MYAPP1+UNDESIG = everyone.
Assume XTCSIZE is ENABLED.

When an UNDESIG group sends a signal, both _XCFMGD and BADGUY1 are candidate classes.
BADGUY1 could be selected, and likely will be if BADGUY1 has paths and CLASSLEN(4028) is a best fit.

So don’t make the mistake of thinking that all UNDESIG signals must flow through _XCFMGD.

Transport Classes when XTCSIZE is DISABLED

Traditional segregation rules



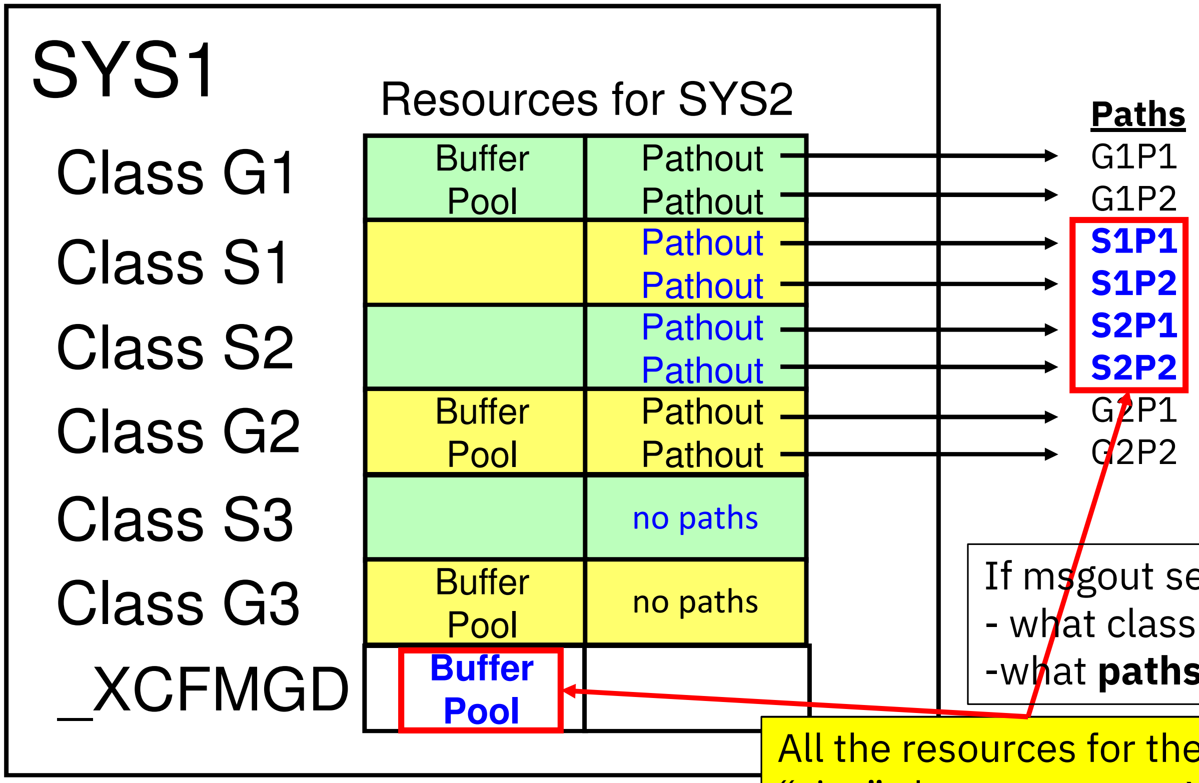
If msgout selects **send** class:
- what class provides **buffer**?
-what **paths** can be used?

Send	**Buffer**	**Paths**
G1	G1	G1Pj
S1	S1	S1Pj
S2	S2	S2Pj
G2	G2	G2Pj
S3	S3	SiPj
G3	G3	(?)

Class Gi segregates by group
Class Si segregates by size (only)

(?) Depends on group assignments:
- If intersects Gi, then GiPj
- Else SiPj

Transport Classes when XTCSIZE is ENABLED



XCF Managed segregation rules

Signals that qualify for classes S1, S2, or S3:

- Get buffer from _XCFMGD
- Can use ANY path assigned to classes S1, S2, or S3

If msgout selects **send** class:
- what class provides **buffer**?
-what **paths** can be used?

Send	Buffer	Paths
G1	G1	G1Pj
S1	MGD	SiPj
S2	MGD	SiPj
G2	G2	G2Pj
S3	MGD	SiPj
G3	G3	(?)

(?) Depends on group assignments

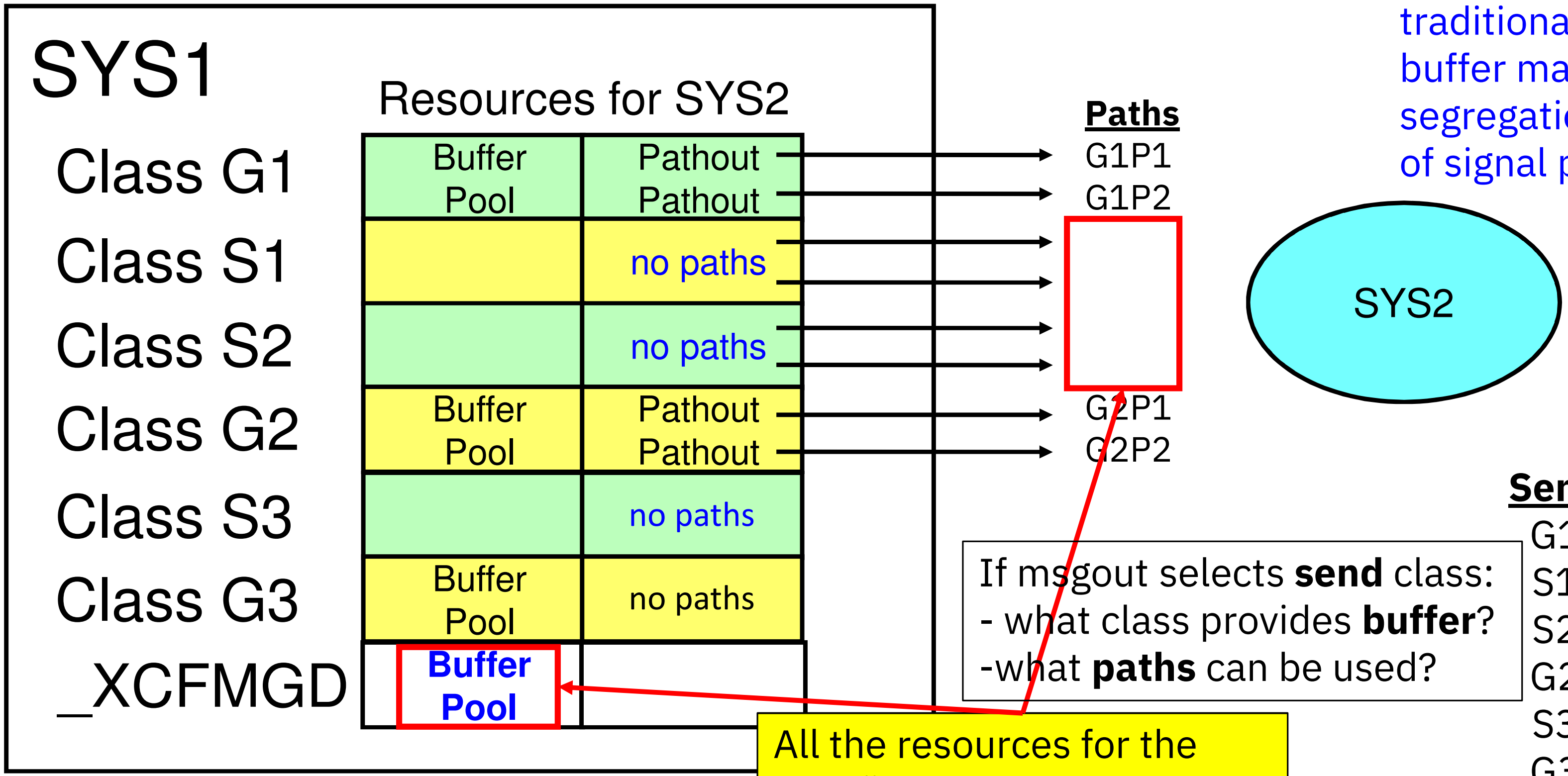
All the resources for the “size” classes are now XCF Managed as one pool of buffers and paths.

Class Gi segregates by group
Class Si segregates by size (only)

Transport Classes when XTCSIZE is ENABLED

Degenerate case. Just as for traditional classes, you get buffer management, but no real segregation due to lack of signal paths.

Health Check



If msgout selects **send** class:
- what class provides **buffer**?
-what **paths** can be used?

All the resources for the "size" classes are now XCF Managed as one pool of buffers (oops, no paths).

Send	**Buffer**	**Paths**
G1	G1	G1Pj
S1	MGD	GiPj
S2	MGD	GiPj
G2	G2	G2Pj
S3	MGD	GiPj
G3	G3	(?)

(?) Depends on group assignments

Class Gi segregates by group
Class Si segregates by size (only)

About the _XCFMGD pseudo transport class

- Uses “best fit” buffers on the send side
  - Maximizes number of signals that can be accepted for a given MAXMSG limit
    - Which is important for handling bursts and delays
  - Traditional classes generally use the defined size which might not be best fit
    - So could encounter “no buffer” condition sooner than with _XCFMGD
- Paths run at the maximum signal size
  - So any size signal can be transmitted without any additional overhead
    - Never any need to re-negotiate (or tune) the paths
  - But that implies buffers on target system are likely bigger than needed
    - Which raises “no buffer” concerns (resiliency, capacity)

Oversize inbound buffers?

- You can actually sustain high signal rates with very few buffers, provided those buffers are recycled quickly
  - For a very long time, we’ve been doing “buffer migration” to reclaim our buffers when member message delivery takes “too long” to free them
  - As of z/OS V2R2, “Message Isolation” allows us to do “aggressive migration” to reclaim more buffers more quickly
  - With z/OS V2R4, we now have “expeditious buffer replenishment” to eliminate much of the latency that elongated “no buffer” resolution
- So we believe we can now sustain high signal rates despite the potential for having fewer inbound buffers for given MAXMSG

No significant increase in “no buffer” conditions

DISABLED

```
                  INBOUND TO    CB88
---------------------------------------------------
         T FROM/TO
FROM     Y DEVICE, OR          REQ BUFFERS TRANSFER
SYSTEM   P STRUCTURE            IN UNAVAIL     TIME
CB86     S IXCPLEX_PATH1     1060K     210    0.156
         S IXCPLEX_PATH2        50       0    0.313
         S IXCPLEX_PATH3     1337K     289    0.139
         S IXCPLEX_PATH4       151       0    0.190
         C C580 TO C564     62,268       0    0.196
         C C581 TO C565     61,871   1,769    0.179
         C C582 TO C566        624       0    0.227
```

2,521,964 REQ IN
2,268 BUFFERS UNAVAIL

ENABLED

```
CB86     S IXCPLEX_PATH1       109       0    0.310
         S IXCPLEX_PATH2     3,220      93    0.280
         S IXCPLEX_PATH3     1767K   2,333    0.131
         S IXCPLEX_PATH4     1,985       0    0.143
         C C580 TO C564      5,897       0    0.173
         C C581 TO C565     85,084       0    0.179
         C C582 TO C566     70,074     209    0.202
```

1,933,369 REQ IN
2,635 BUFFERS UNAVAIL

Yes, “no buffer” increased.
But certainly not to the extent one
might expect based on relative buffer sizes.
And not enough to impact throughput.

Oversize inbound buffers - implications

- More buffer space consumed by the normal steady state working set (likely fewer in number, but they are larger)
  - Inbound list paths will drop to zero buffers in use if there is not enough traffic to keep them busy.
  - Inbound CTC devices generally have at least 4 buffers always in use.
- But may not need as much space for the peaks
  - We will still consume up to the MAXMSG limit you provide
  - For those that have been increasing inbound MAXMSG to avoid “no buffer” conditions, you may not need such large values
    - Indeed, large values can induce long queue effects
    - In some of our tests, throughput degraded as MAXMSG increased!

Inbound “no buffer” conditions

- Our use of large buffers implies “fewer” buffers for a given MAXMSG value, which increases potential for “no buffer”
- Historically, “no buffer” seems to incent people to increase their MAXMSG values, sometimes to unreasonably high values
    - Generally, if you hit MAXMSG limit, you have queueing. The bigger the MAXMSG limit that you hit, the more queueing you have.
    - Driving “no buffer” to zero looks nice, but that queue containing tens of thousands of signals could be more detrimental to system performance than the “no buffer” condition itself
- On the inbound side, “no buffer” is not necessarily bad ....

Inbound “no buffer” counts

- For a CTC device, I/O is started in anticipation of there being a signal to receive. If “no buffer”, we could not start the desired I/O.
  - But if there is nothing to receive, there is no signal transfer delay.
  - So long as we get a buffer and start the I/O before a signal is sent, there is no impact no matter how high the “no buffer” count climbs
- For a list path, I/O is started when we believe there is a signal to receive. So a “no buffer” condition likely implies delay.
  - But not necessarily. Sometimes our belief is unfounded.
- XCF abhors inbound “no buffer” conditions, so it repeatedly looks to resolve them, which can drive up the “no buffer” count.
- But the count tells us very little about the impact on signal transfer since we can’t tell how much delay was induced by the “no buffer” condition.

Measuring impactful “no buffer” conditions

- We are providing additional inbound path measurements to capture the notion of an “impactful no buffer condition”
  - If we hit a “no buffer” condition, did it impact signal transfer?
  - If so, how long did the impact last?
- To some degree, we wanted insight as to whether our use of largest signal buffers (and therefore smaller number) was inducing signal transfer delays.
- But more generally, we anticipate this data will inform judgments about how many signal paths are needed:
  - Maybe we need more buffers instead of a new path
  - Maybe we need buffers to get recycled more quickly
  - Maybe we don’t care because the impacts are inconsequential

XCF defines the _XCFMGD pseudo transport class

- You cannot directly modify its attributes. However:
  - Buffer space MAXMSG limit used for _XCFMGD takes into account:
    - Default MAXMSG value
    - Buffer space limits of the classes that are being managed
    - Our own judgment as to what is a reasonable value
  - So changes to default MAXMSG value, or MAXMSG values for a managed class or a managed path could cause the buffer limits for _XCFMGD to change
    - Though you might need to DISABLE and then ENABLE the XTCSIZE switch to have them take effect
- You cannot directly assign paths to the _XCFMGD class
  - Paths are “inherited” from the classes managed when XTCSIZE is ENABLED
  - So you continue to assign paths to a traditional transport class
  - But they are reported as _XCFMGD when XTCSIZE is ENABLED
    - This lets you “see” the new behavior

DISPLAY XCF accepts _XCFMGD as a class name.
SETXCF and COUPLExx do not.

What will you see?

DISPLAY XCF output

RMF reports of XCF activity

D XCF,C – What is the XTCSIZE switch setting?

```
D XCF,C
IXC357I  16.08.07  DISPLAY XCF
SYSTEM SY1 DATA
....
    OPTIONAL FUNCTION STATUS:
     FUNCTION NAME                   STATUS      DEFAULT
     DUPLEXCF16                      DISABLED    DISABLED
     SYSSTATDETECT                   ENABLED     ENABLED
     USERINTERVAL                    ENABLED     DISABLED
     CRITICALPAGING                  DISABLED    DISABLED
     DUPLEXCFDIAG                    DISABLED    DISABLED
     CFLCRMGMT                       DISABLED    DISABLED
     COUPLINGTHININT                 ENABLED     ENABLED
     CFSTRQMON                       DISABLED    DISABLED
     MSGISO                          ENABLED     ENABLED
     XTCSIZE                         ENABLED     ENABLED
 ....
```

XCF Managed is active on this system.

When sending signals to a target system running z/OS V2R4, the signal resources for all of the “size only” transport classes are being XCF Managed as a single pool that supports any size signal.

When sending signals to a down level target system, traditional transport classes and traditional segregation rules are used.

D XCF,C – What is the XTCSIZE switch setting?

```
D XCF,C
IXC357I  16.08.07  DISPLAY XCF
SYSTEM SY1 DATA
....
    OPTIONAL FUNCTION STATUS:
     FUNCTION NAME                   STATUS      DEFAULT
     DUPLEXCF16                      DISABLED    DISABLED
     SYSSTATDETECT                   ENABLED     ENABLED
     USERINTERVAL                    ENABLED     DISABLED
     CRITICALPAGING                  DISABLED    DISABLED
     DUPLEXCFDIAG                    DISABLED    DISABLED
     CFLCRMGMT                       DISABLED    DISABLED
     COUPLINGTHININT                 ENABLED     ENABLED
     CFSTRQMON                       DISABLED    DISABLED
     MSGISO                          ENABLED     ENABLED
     XTCSIZE                         DISABLED    ENABLED
 ....
```

XCF Managed is not active.

You are using traditional transport classes and traditional transport class segregation rules.

Change XTCSIZE switch setting - dynamically

```
SETXCF FUNCTIONS,DISABLE=XTCSIZE
IXC373I XCF / XES OPTIONAL FUNCTIONS DISABLED:
        XTCSIZE
```

Revert to old behavior

```
SETXCF FUNCTIONS,ENABLE=XTCSIZE
IXC373I XCF / XES OPTIONAL FUNCTIONS ENABLED:
        XTCSIZE
```

Enable new behavior

Change XTCSIZE switch setting – through IPL

COUPLExx parmlib member

```
COUPLE   SYSPLEX(plexname)
   PCOUPLE(…)
   ACOUPLE(…)
   ….
FUNCTIONS ENABLE(XTCSIZE)
CLASSDEF ….
PATHIN  ….
PATHOUT ….
```

Enable new behavior

COUPLExx parmlib member

```
COUPLE   SYSPLEX(plexname)
   PCOUPLE(…)
   ACOUPLE(…)
   ….
FUNCTIONS DISABLE(XTCSIZE)
CLASSDEF ….
PATHIN  ….
PATHOUT ….
```

Revert to old behavior

D XCF,CLASSDEF – to see transport class data

```
D XCF,CLASSDEF,CLASS=ALL
IXC344I  16.08.23  DISPLAY XCF 848
   TRANSPORT     CLASS        DEFAULT     ASSIGNED
   CLASS         LENGTH       MAXMSG      GROUPS
   _XCFMGD            0         2000      UNDESIG
   BADGUY0          956         2000      MYAPP1
   BADGUY1         4028         2000      MYAPP1    UNDESIG
   BADGUY2         8124         2000      UNDESIG
   BIG            32000         2000      UNDESIG
   DEFAULT          956         2000      UNDESIG
   MED            12000         2000      UNDESIG
   SML              956         2000      UNDESIG
```

Always present

These classes selected when XTCSIZE is ENABLED.

We don’t have anything to show you which classes qualify for XTCSIZE management.

D XCF,CLASSDEF – to see transport class data (cont)

As for any traditional class, _XCFMGD signal size distributions are shown for each target system.

```
….
_XCFMGD TRANSPORT CLASS USAGE FOR SYSTEM SY1
SUM MAXMSG:      10000    IN USE:          2  NOBUFF:          0
  SEND CNT:        978  BUFFLEN (FIT):   956
  SEND CNT:        362  BUFFLEN (BIG):  4028
  SEND CNT:         21  BUFFLEN (BIG):  8124

_XCFMGD TRANSPORT CLASS USAGE FOR SYSTEM SY2
SUM MAXMSG:      22000    IN USE:          8  NOBUFF:          0
  SEND CNT:       1713  BUFFLEN (FIT):   956
  SEND CNT:        130  BUFFLEN (BIG):  4028
  SEND CNT:        119  BUFFLEN (BIG):  8124

_XCFMGD TRANSPORT CLASS USAGE FOR SYSTEM SY3
SUM MAXMSG:      16000    IN USE:          0  NOBUFF:          0
  SEND CNT:          0  BUFFLEN (FIT):   956
```

SY1 is local system

SY2 running z/OS V2R4
(“XCF Managed” is in play)

SY3 running z/OS V2R3
Down level, so old rules.
(“XCF Managed” not used)

Which list paths are XCF Managed ?

```
D XCF,PO,STRNAME=ALL,STATUS=WORKING
IXC356I  16.09.28  DISPLAY XCF 855
STRNAME         REMOTE   PATHOUT     UNUSED             TRANSPORT
PATHOUT         SYSTEM   STATUS      PATHS    RETRY  MAXMSG CLASS
IXCTL_SIGNAL01           WORKING       234       10    2000 DEFAULT
                SY2      WORKING                            _XCFMGD
                SY3      WORKING                            DEFAULT
IXCTL_SIGNAL02           WORKING       234       10    2000 DEFAULT
                SY2      WORKING                            _XCFMGD
                SY3      WORKING                            DEFAULT

STRNAME         REMOTE   PATHOUT    TRANSFR BUFFER  MSGBUF SIGNL MXFER
PATHOUT    LIST SYSTEM   STATUS     PENDING LENGTH  IN USE NUMBR TIME
IXCTL_SIGNAL01
             12 SY2      WORKING          0  62464     204    10  2915
              9 SY3      WORKING          0    956       6     3  4612
IXCTL_SIGNAL02
             12 SY2      WORKING          0  62464     270    11  2767
              9 SY3      WORKING          0    956       8     4  1049
```

Structure still indicates the “home” class

Uplevel target system

Downlevel target system (so using old rules)

Running at biggest size

Running at defined size

Same list paths after XTCSIZE is DISABLED

```
D XCF,PO,STRNAME=ALL,STATUS=WORKING
IXC356I  16.10.32  DISPLAY XCF 871
STRNAME         REMOTE   PATHOUT     UNUSED                TRANSPORT
PATHOUT         SYSTEM   STATUS      PATHS   RETRY  MAXMSG CLASS
IXCTL_SIGNAL01           WORKING       234      10    2000 DEFAULT
                SY2      WORKING                           DEFAULT
                SY3      WORKING                           DEFAULT
IXCTL_SIGNAL02           WORKING       234      10    2000 DEFAULT
                SY2      WORKING                           DEFAULT
                SY3      WORKING                           DEFAULT

STRNAME         REMOTE   PATHOUT    TRANSFR BUFFER  MSGBUF SIGNL MXFER
PATHOUT   LIST  SYSTEM   STATUS     PENDING LENGTH  IN USE NUMBR TIME
IXCTL_SIGNAL01
            12  SY2      WORKING          0    956      12    18  3418
             9  SY3      WORKING          0    956       8     4  3044
IXCTL_SIGNAL02
            12  SY2      WORKING          0    956      14    19   262
             9  SY3      WORKING          0    956      10     5  3116
```

No class is XCF Managed

Reverts to “home” class

Reverts to defined size

Which CTC signal paths are XCF Managed?

DEFAULT class is XCF Managed.
BADGUY0 is not.

```
D XCF,PO,DEV=ALL,STATUS=WORKING
IXC356I  16.09.14  DISPLAY XCF 852
LOCAL DEVICE     REMOTE    PATHOUT      REMOTE                TRANSPORT
PATHOUT          SYSTEM    STATUS       PATHIN   RETRY  MAXMSG CLASS
8000             SY2       WORKING      80BF        10    2000 BADGUY0
8001             SY2       WORKING      80BE        10    2000 _XCFMGD
8002             SY3       WORKING      80BD        10    2000 BADGUY0
8003             SY3       WORKING      80BC        10    2000 DEFAULT

LOCAL   REMOTE  REMOTE    PATHOUT     TRANSFR BUFFER  MSGBUF SIGNL MXFER
PATHOUT PATHIN  SYSTEM    STATUS      PENDING LENGTH  IN USE NUMBR TIME
8000    80BF    SY2       WORKING           0    956      14   826    34
8001    80BE    SY2       WORKING           0  62464      14  1387    38
8002    80BD    SY3       WORKING           0    956      14  1193    48
8003    80BC    SY3       WORKING           0    956      14  1847    29
```

You can't see the "home" class

Uplevel target system

Downlevel target system
(so using old rules)

Running at biggest size

Running at defined size

Same CTC paths after XTCSIZE is DISABLED

No class is XCF Managed

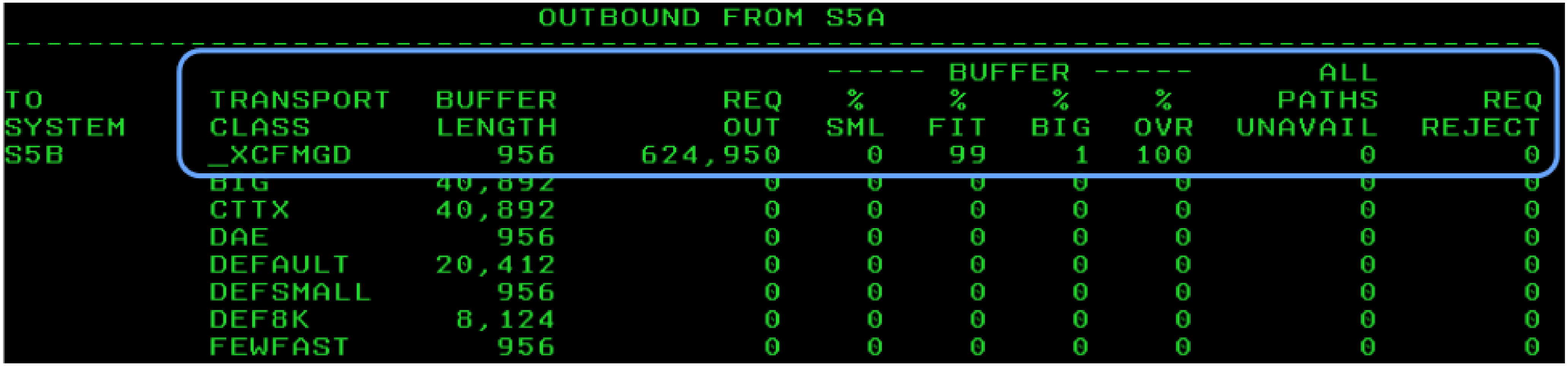
```
D XCF,PO,DEV=ALL,STATUS=WORKING
IXC356I  16.10.28  DISPLAY XCF 868
LOCAL DEVICE     REMOTE   PATHOUT      REMOTE                TRANSPORT
PATHOUT          SYSTEM   STATUS       PATHIN   RETRY  MAXMSG CLASS
8000             SY2      WORKING      80BF        10    2000 BADGUY0
8001             SY2      WORKING      80BE        10    2000 DEFAULT
8002             SY3      WORKING      80BD        10    2000 BADGUY0
8003             SY3      WORKING      80BC        10    2000 DEFAULT

LOCAL   REMOTE  REMOTE   PATHOUT      TRANSFR BUFFER  MSGBUF SIGNL MXFER
PATHOUT PATHIN  SYSTEM   STATUS       PENDING LENGTH  IN USE NUMBR TIME
8000    80BF    SY2      WORKING            0    956      14   828    35
8001    80BE    SY2      WORKING            0    956      14  1497    36
8002    80BD    SY3      WORKING            0    956      14  1194    29
8003    80BC    SY3      WORKING            0    956      14  1950    23
```

Reverts to “home” class

Reverts to defined size

RMF Report: XCF Usage by System



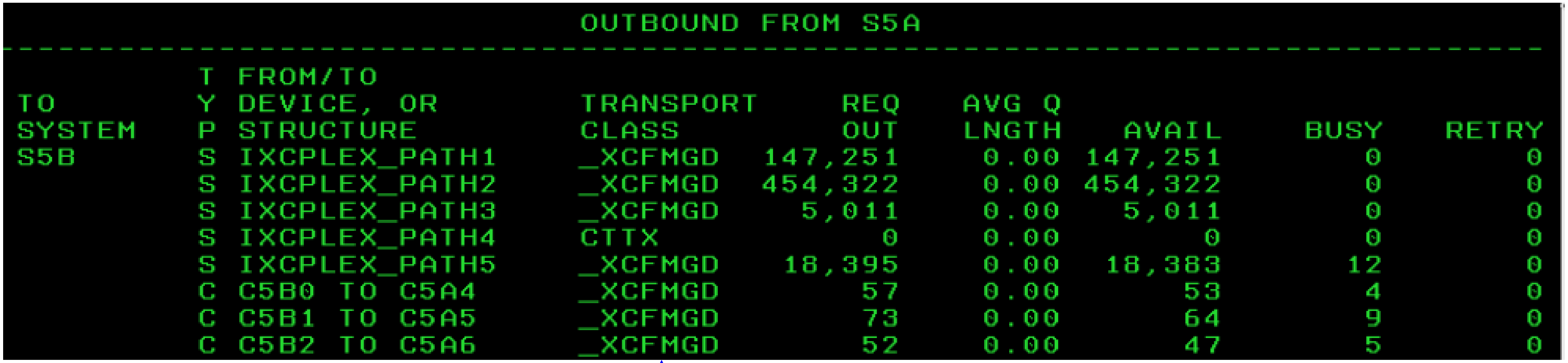
```
                             OUTBOUND FROM S5A
--------------------------------------------------------------------------------
                                                ----- BUFFER -----        ALL
TO         TRANSPORT  BUFFER          REQ     %     %     %     %       PATHS      REQ
SYSTEM     CLASS      LENGTH          OUT   SML   FIT   BIG   OVR     UNAVAIL   REJECT
S5B        _XCFMGD       956      624,950     0    99     1   100           0        0
           BIG        40,892            0     0     0     0     0           0        0
           CTTX       40,892            0     0     0     0     0           0        0
           DAE           956            0     0     0     0     0           0        0
           DEFAULT    20,412            0     0     0     0     0           0        0
           DEFSMALL      956            0     0     0     0     0           0        0
           DEF8K       8,124            0     0     0     0     0           0        0
           FEWFAST       956            0     0     0     0     0           0        0
```

_XCFMGD transport class will always appear.

- Its counts will be zero if XTCSIZE is DISABLED or if the target system is down level.
- SML, FIT, BIG are truthful and consistent with existing definitions of SMF data.
  In the past, these numbers might have prompted one to consider revising class definitions.
  But with _XCFMGD, not only is there nothing to be done, there is nothing you can do.

The traditional classes will always appear as well. The counts will be zero if the class is not being used.
But from this report, you can't tell why there's no use: XCF Managed? No signals sent? (CTTX for example).

RMF Report: XCF Path Statistics



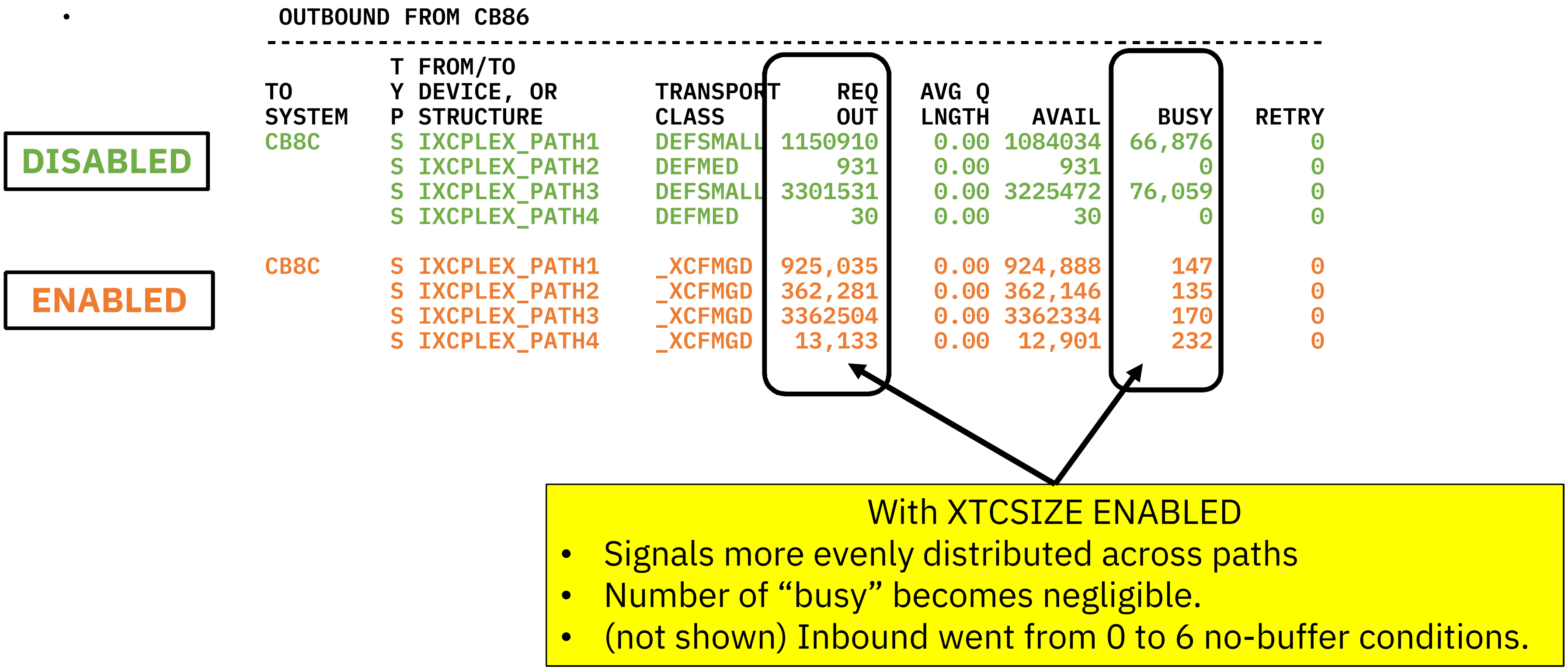
```
                             OUTBOUND FROM S5A
-----------------------------------------------------------------------------
         T FROM/TO
TO       Y DEVICE, OR        TRANSPORT     REQ    AVG Q
SYSTEM   P STRUCTURE         CLASS         OUT    LNGTH    AVAIL     BUSY    RETRY
S5B      S IXCPLEX_PATH1     _XCFMGD   147,251     0.00  147,251        0        0
         S IXCPLEX_PATH2     _XCFMGD   454,322     0.00  454,322        0        0
         S IXCPLEX_PATH3     _XCFMGD     5,011     0.00    5,011        0        0
         S IXCPLEX_PATH4     CTTX            0     0.00        0        0        0
         S IXCPLEX_PATH5     _XCFMGD    18,395     0.00   18,383       12        0
         C C5B0 TO C5A4      _XCFMGD        57     0.00       53        4        0
         C C5B1 TO C5A5      _XCFMGD        73     0.00       64        9        0
         C C5B2 TO C5A6      _XCFMGD        52     0.00       47        5        0
```

Paths are reported as being assigned to _XCFMGD transport class if they are being XCF Managed.

What might you see

- Most of you likely have:
  - Multiple “pure size” transport classes defined
  - One or more signal paths assigned to each of those classes
  - Very high AVAIL percentages for your paths
    - Signal paths nearly always “not busy” when picked to send a signal
    - Some paths hardly ever picked, so lots of excess capacity
  - Hardly any “no buffer” conditions
- So when XTCSIZE is ENABLED
  - Best fit buffers for _XCFMGD likely implies more send side buffer capacity than your definitions provide
  - Signals likely distributed to paths much as your traditional classes do today
  - An underutilized path in one of your classes can now be used to help a peer class that otherwise would have used a “busy” path

Signal distributions might change



```
 OUTBOUND FROM CB86
----------------------------------------------------------------------------
         T FROM/TO
TO       Y DEVICE, OR       TRANSPORT    REQ   AVG Q
SYSTEM   P STRUCTURE        CLASS        OUT   LNGTH   AVAIL    BUSY   RETRY
CB8C     S IXCPLEX_PATH1    DEFSMALL 1150910    0.00 1084034  66,876       0
         S IXCPLEX_PATH2    DEFMED       931    0.00     931       0       0
         S IXCPLEX_PATH3    DEFSMALL 3301531    0.00 3225472  76,059       0
         S IXCPLEX_PATH4    DEFMED        30    0.00      30       0       0

CB8C     S IXCPLEX_PATH1    _XCFMGD  925,035    0.00 924,888     147       0
         S IXCPLEX_PATH2    _XCFMGD  362,281    0.00 362,146     135       0
         S IXCPLEX_PATH3    _XCFMGD  3362504    0.00 3362334     170       0
         S IXCPLEX_PATH4    _XCFMGD   13,133    0.00  12,901     232       0
```

With XTCSIZE ENABLED

- Signals more evenly distributed across paths
- Number of “busy” becomes negligible.
- (not shown) Inbound went from 0 to 6 no-buffer conditions.

Expectations

- Overall, your sysplex workloads should run at least as well as they do now. Signal performance should not be degraded.
  - Regardless of what z/OS release the target system is running
  - Regardless of whether XTCSIZE is ENABLED or DISABLED
- For a given set of signal paths, there is no set of (pure size) transport class definitions for those paths that will perform better with XTCSIZE=DISABLED than with XTCSIZE=ENABLED.
- The only determining factors for signal performance will be:
  - The number of signal paths you provide
  - The performance characteristics of those paths
  - The performance characteristics of the systems using those paths

Possible exceptions?

- Our overall improvements to signal delivery might change timings that induce secondary impacts
- Signal traffic might flow over different paths than in the past, which might impact things like CF, subchannel, and link utilization
  - If you have classes with “busy” paths today, those signals are likely to be sent via an available path from a peer XCF Managed class.
  - So maybe your high frequency signals always traveled via CF1. With XTCSIZE=ENABLED, any signal can go via any XCF Managed path. So CF2, which in the past got very little activity, might now be used more frequently.
- Use of max size inbound buffers might be an issue for installations that are memory constrained and/or have lots of signal paths (especially CTC devices)

Session Summary

- We have eliminated the need for you to define XCF Transport Classes to segregate signals by size
  - No more planning, defining, monitoring, tuning, changing, …
- Just ENABLE the XTCSIZE switch. We'll do the rest.
  - In particular, you don't need to change your current XCF transport class configuration

Appendix

- Contacts
  - Mark A Brooks – mabrook@us.ibm.com
  - Neil Johnson - najohnsn@us.ibm.com