E13

IMS Shared Queues Functions and Facilities

Bill Stillwell, Dallas Systems Center

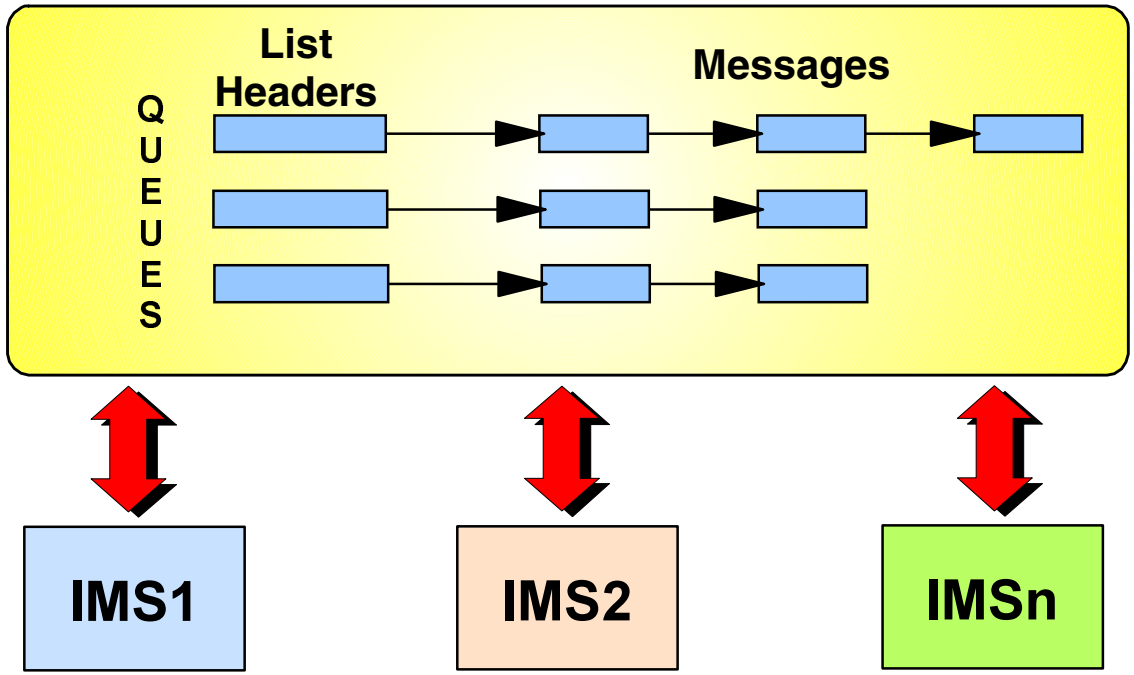
Anaheim, California October 23 - 27, 2000

RETURN TO INDEX

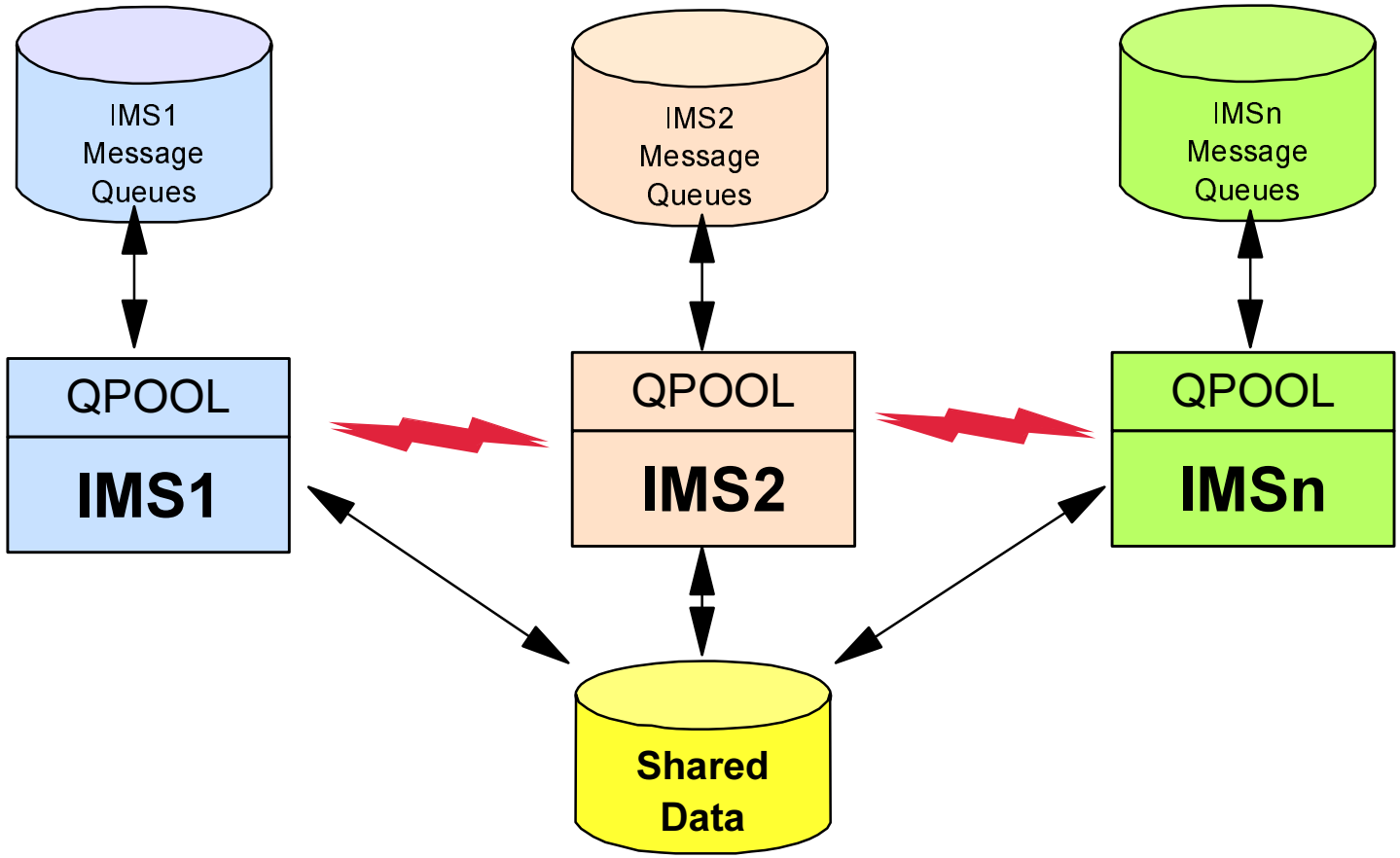
What are Shared Queues?

A set of input and output message queues which can be shared by multiple IMSs in a Parallel Sysplex.

Coupling Facility



Before Shared Queues

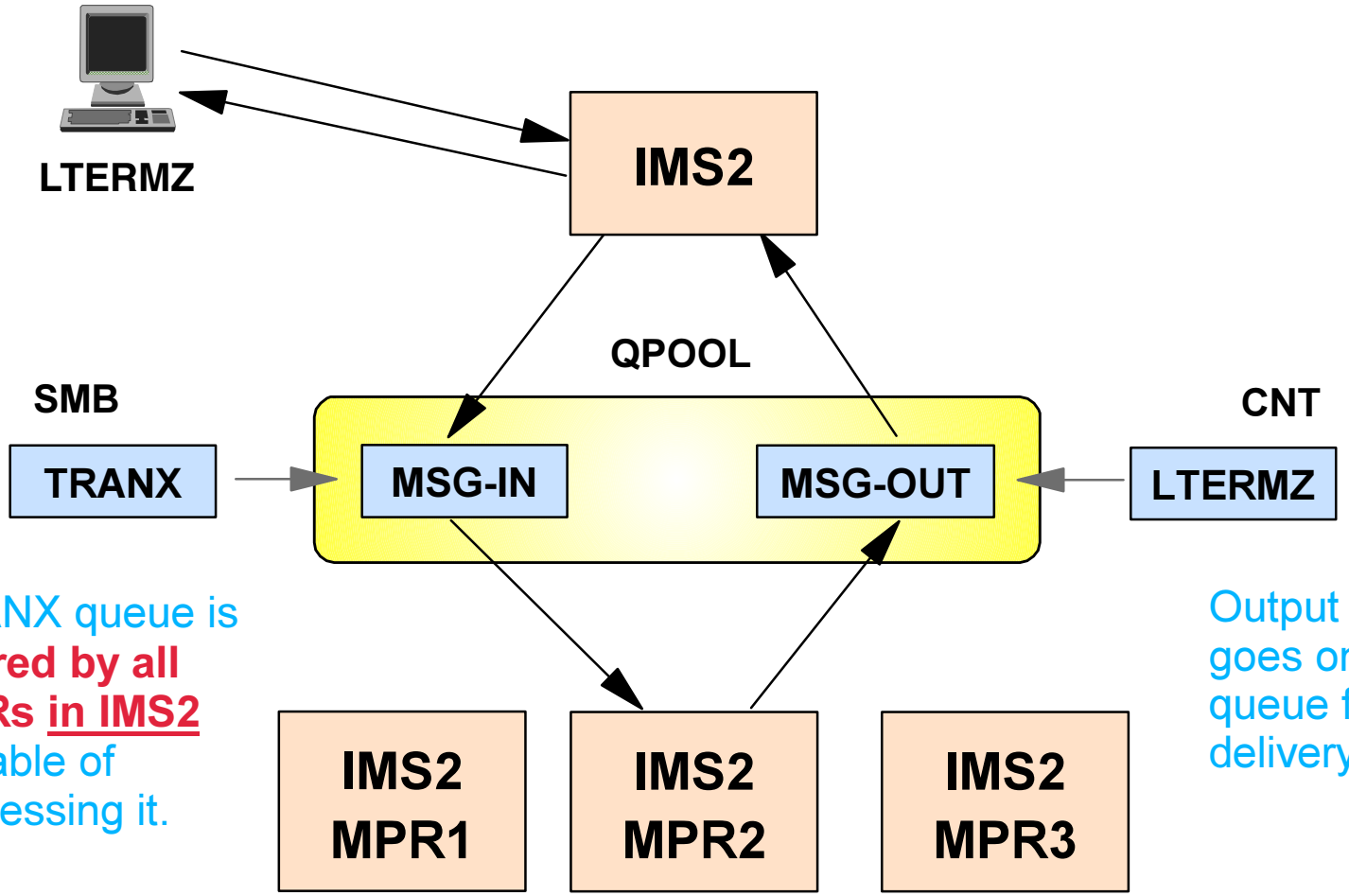


★ IMSs could share the data, but each IMS had *exclusive* use of its own message queues

★ *Application load balancing* among IMS systems was a user responsibility

✓ Network balancing, Workload Router, MSC, ISC, APPC, ...

Before Shared Queues ...

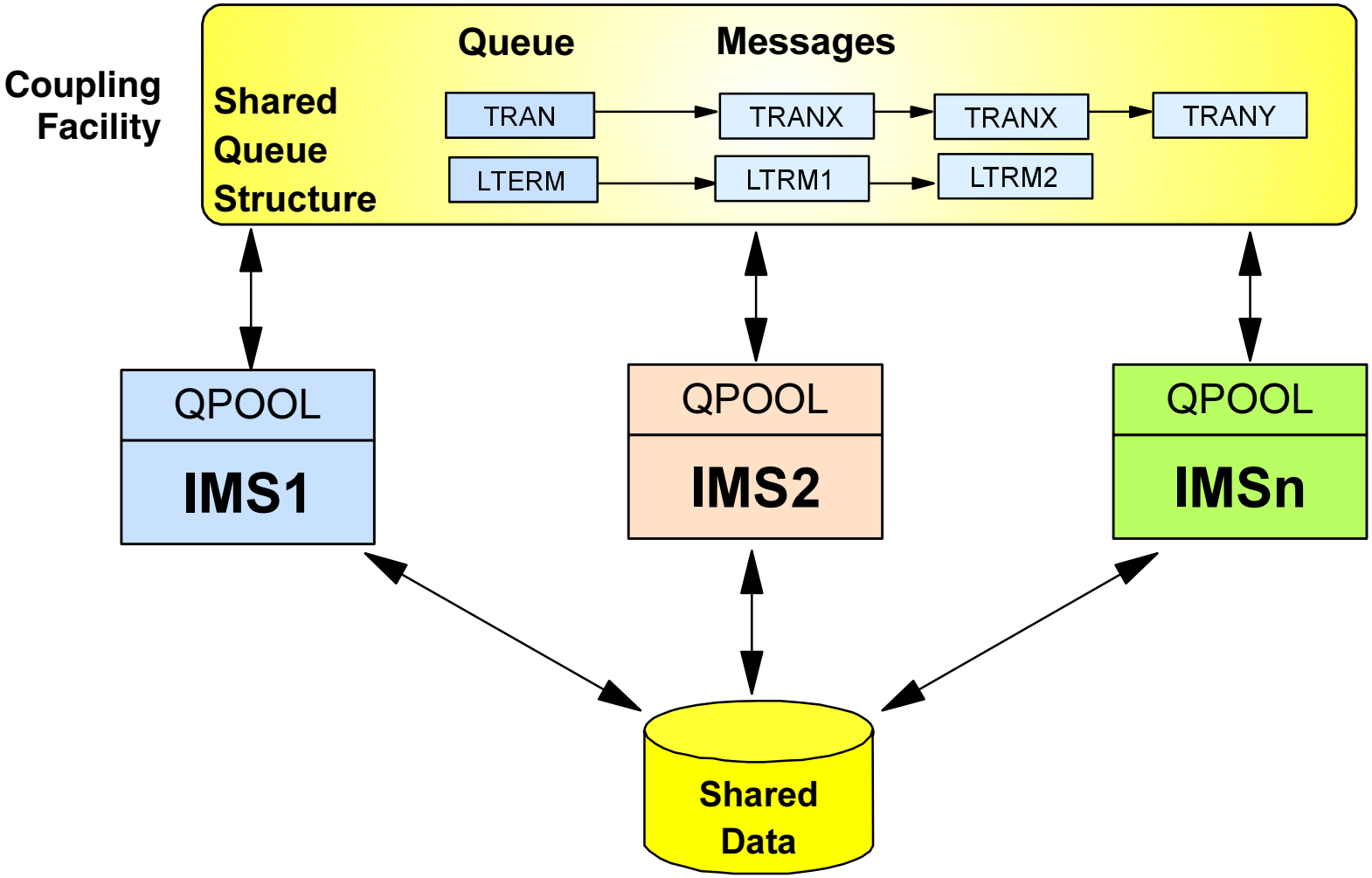


TRANX queue is **shared by all MPRs in IMS2** capable of processing it.

Output message goes on LTERMZ queue for delivery by IMS2.

Message Processing Region 2 is first to ask for TRANX.

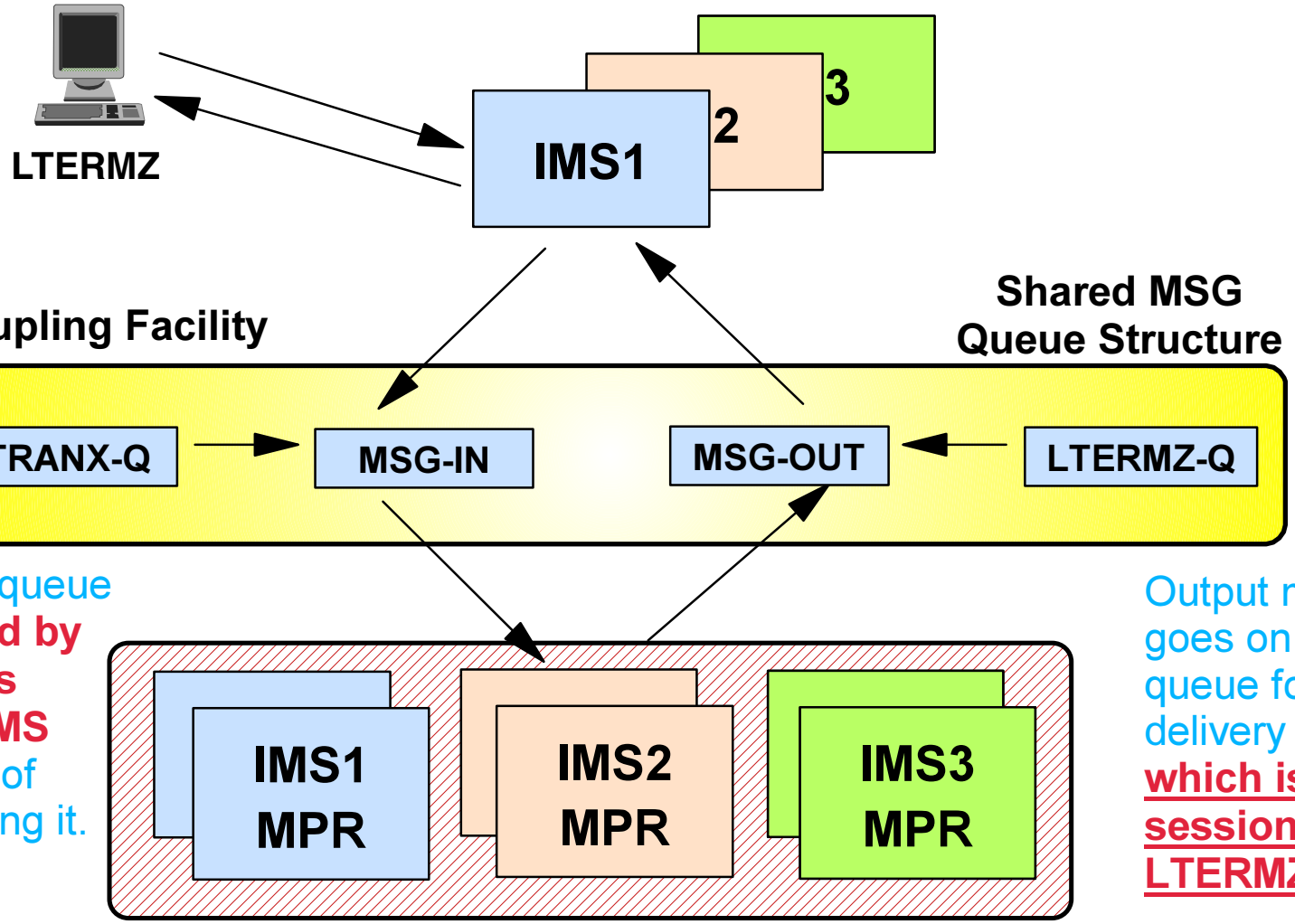
With Shared Queues



★ Individual IMS message queues have been replaced by *Shared Queue Structures* in the Coupling Facility.

★ Messages in the Shared Queue Structures are *available to all IMSs.*

With Shared Queues ...

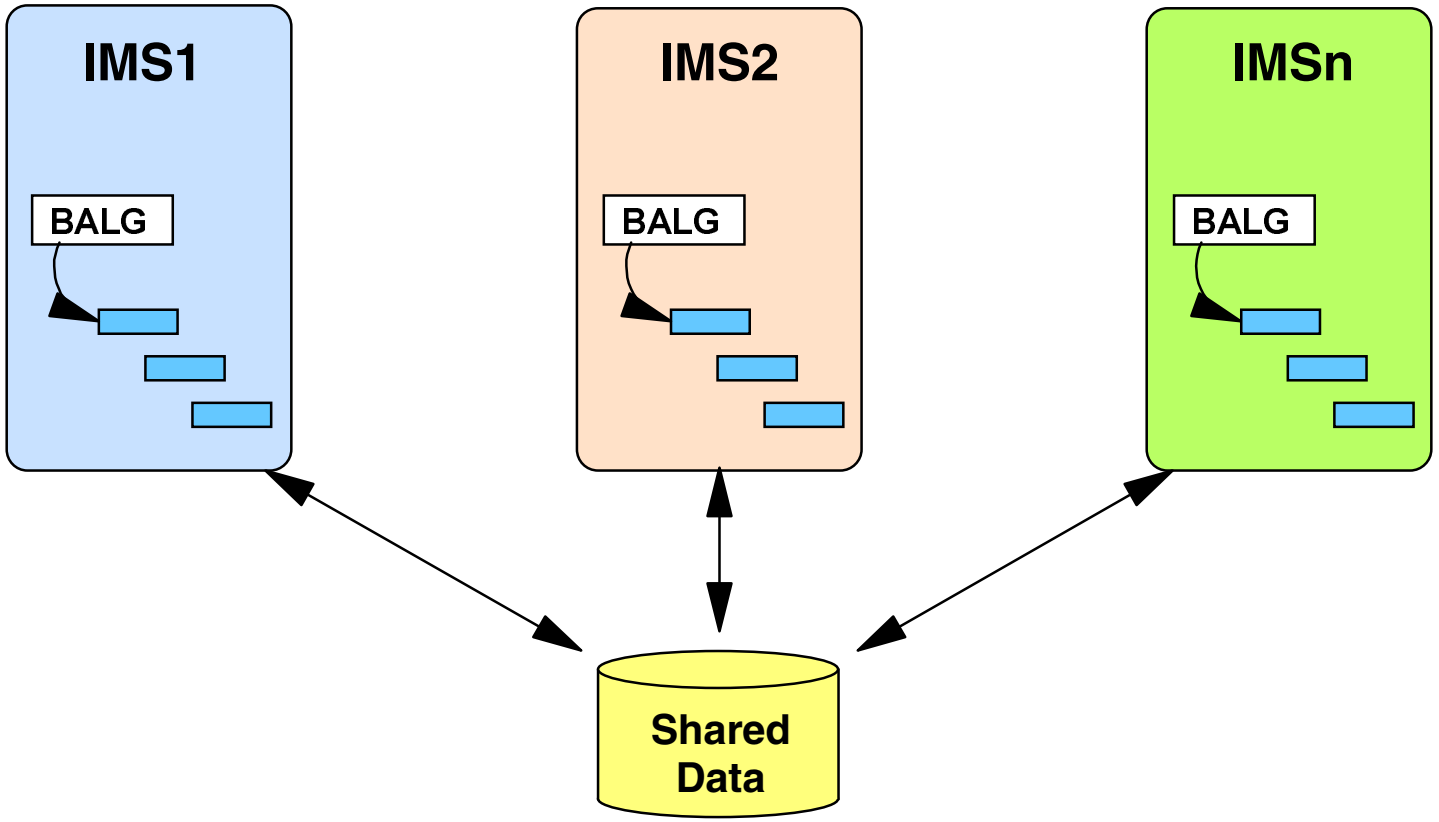


TRANX queue is **shared by all MPRs in any IMS** capable of processing it.

Output message goes on LTERM queue for delivery by IMS **which is in session with LTERMZ**

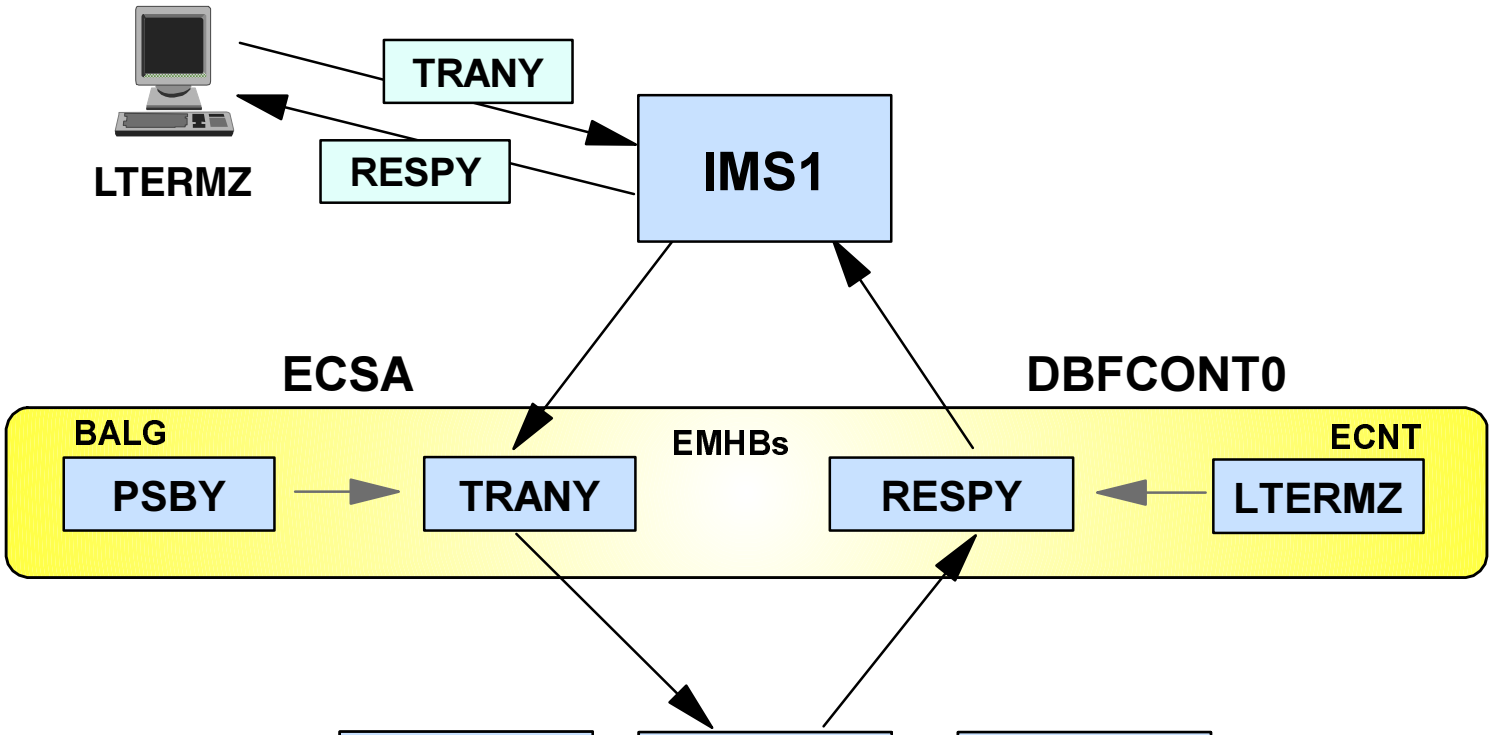
MPR **in IMS2** is first to ask for TRANX

Before Shared EMH



- ★ Each IMS has exclusive access to its own balancing group queues
- ★ Application load balancing among IMS systems is a user responsibility
  - ✓ Network balancing

Before Shared EMH ...



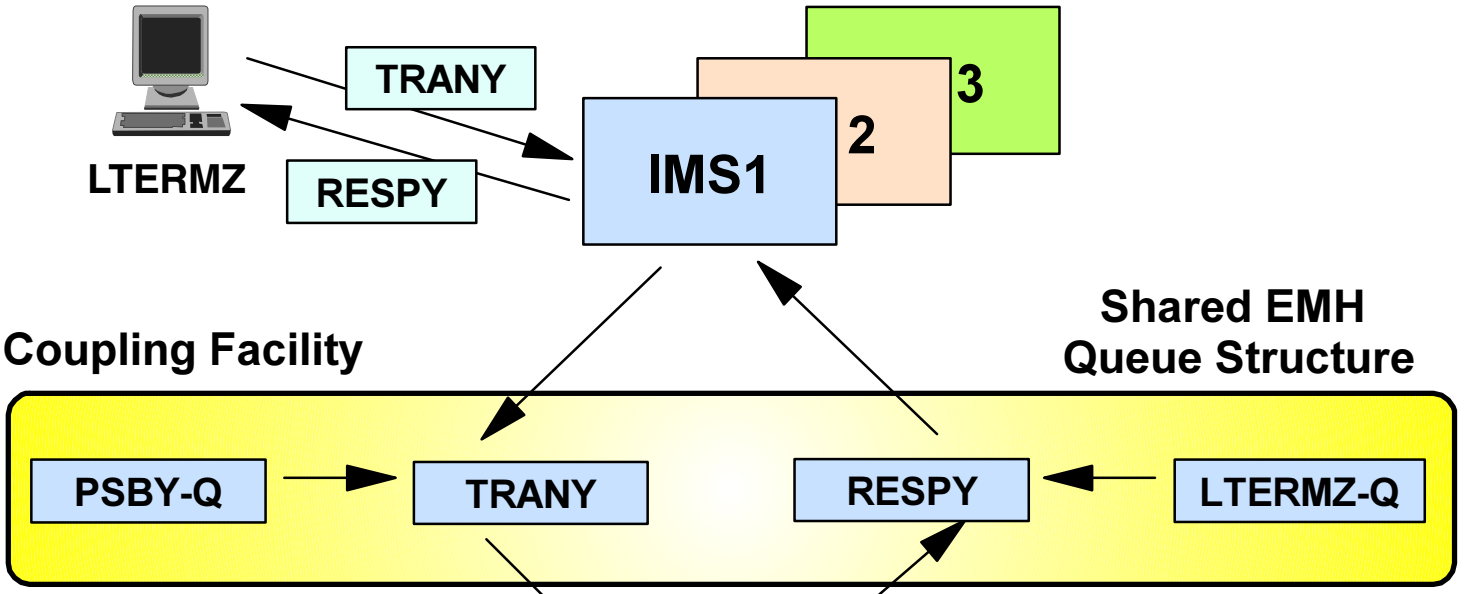
PSBY queue is **shared by all IFPs in IMS1** capable of processing it.

IFP1 PSBY | **IFP2 PSBY** | **IFP3 PSBY**

Output message goes on LTERMZ queue for delivery by IMS.

IFP2 is first to ask for TRANY

With Shared EMH



PSBY queue is **shared by all IFPs in any IMS** capable of processing it.

IFP **in IMS2** is first to ask for TRANY

Output message goes on LTERM queue for delivery by IMS **which is in session with LTERMZ**

Shared Queues

Shared Message Queues (MSGQ)

- IMS full function messages **available to multiple IMS subsystems**
- Multiple input queue types
  - Transaction, Serial Transaction, Suspend
- Multiple output queue types
  - LTERM, APPC-Out, OTMA-Out, Remote (MSC)
- Staging queue types (for long messages)
  - Transaction and LTERM

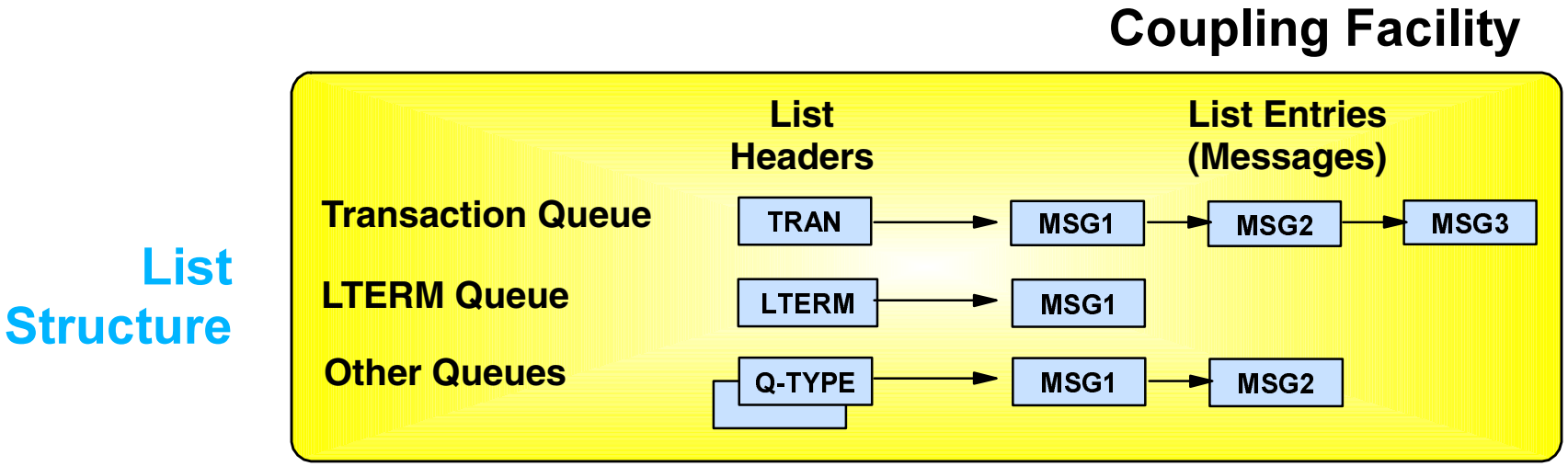
Shared EMH Queues (EMHQ)

- Fast path EMH messages **available to multiple IMS subsystems**
- Single input queue type
  - Program (PSB/BALG)
- Single output queue type
  - LTERM

Shared Queues ...

Queues are maintained in List Structures in the Coupling Facility

- Defined in CFRM Policy
- One primary structure for FF messages
  - Optional overflow structure
- One primary structure for EMH messages
  - Optional overflow structure



Implementation of Shared Queues

IMS

- ***Connects to MSGQ and to EMHQ*** list structures
  - Connection is through a Common Queue Server (CQS)
- ***Registers interest*** in specific queues
  - Indicates that IMS is capable of processing a message on that queue

Input Queues

- MSG input queues (e.g. Transaction Ready Queue for TRANX)
  - TRANX must be defined to IMS and **not stopped**
  - Multiple IMSs may register interest in the same transaction
- EMH input queue (e.g. Program Ready Queue for PSBY)
  - PSBY must be defined to IMS and **not stopped**
  - IFP region with PSBY must be started **somewhere in the Shared Queues Group**
  - Multiple IMSs may register interest in the same PSB

Implementation ...

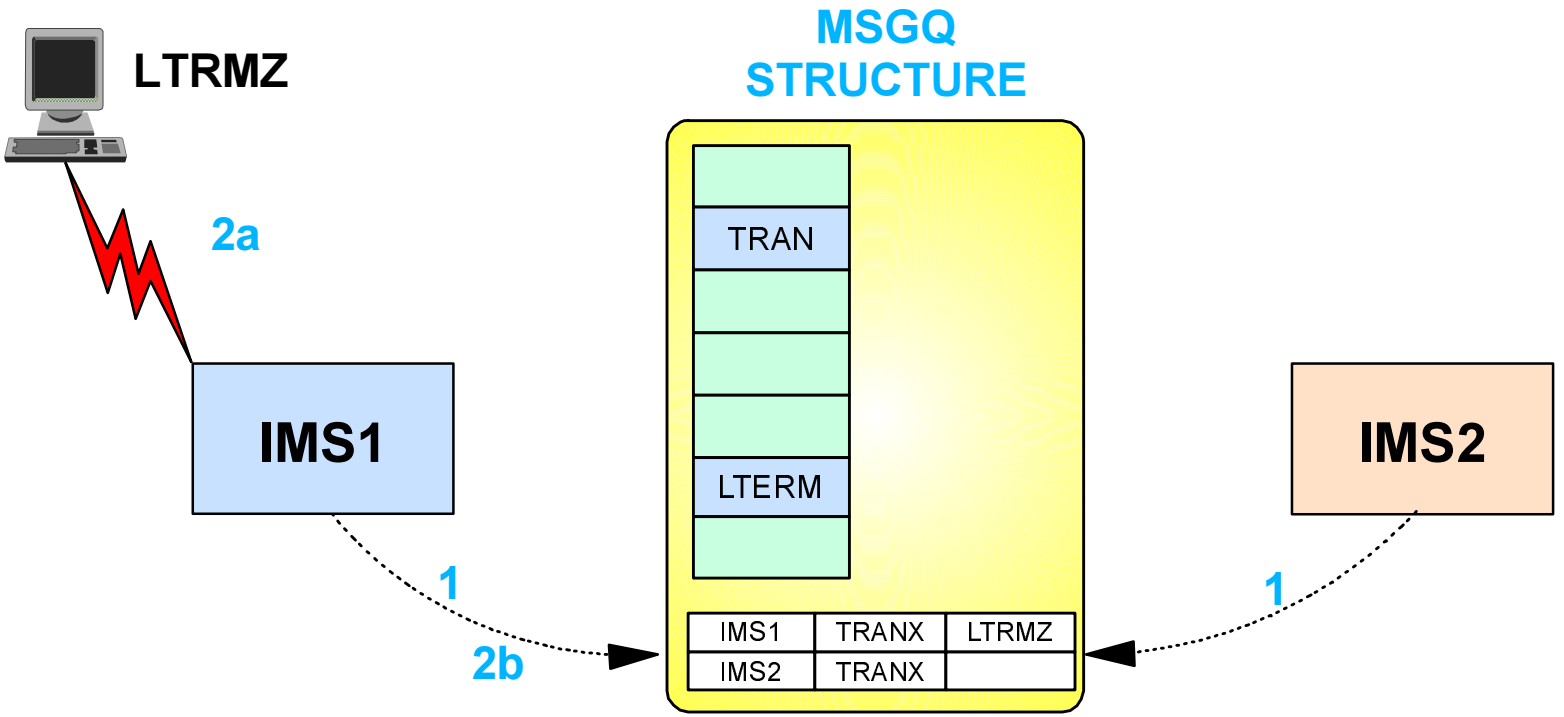
Output queues (e.g. LTERM Ready Queue for LTERMZ)

- LTERMZ must be defined statically or dynamically to IMS
- ***Only one IMS*** in the shared queues group will (should) register interest in the output queue for LTERMZ
  - LTERMZ must be **in session** with IMS to register interest

Message handling

- When IMSn receives an input message it places it on the shared input **Transaction Ready Queue**
  - ***Any IMS with registered interest in the transaction*** may retrieve it from the shared queue and process it
  - May be multiple IMSs with registered interest
- When IMSn has an output message, it places it on the shared output **LTERM Ready Queue**
  - ***Any IMS with registered interest in the LTERM*** may retrieve it from the queue and send it
  - Should be only one IMS with registered interest

Shared Message Queue Example



1. At initialization, IMS1 and IMS2 connect to MSGQ structure and register interest in TRANX

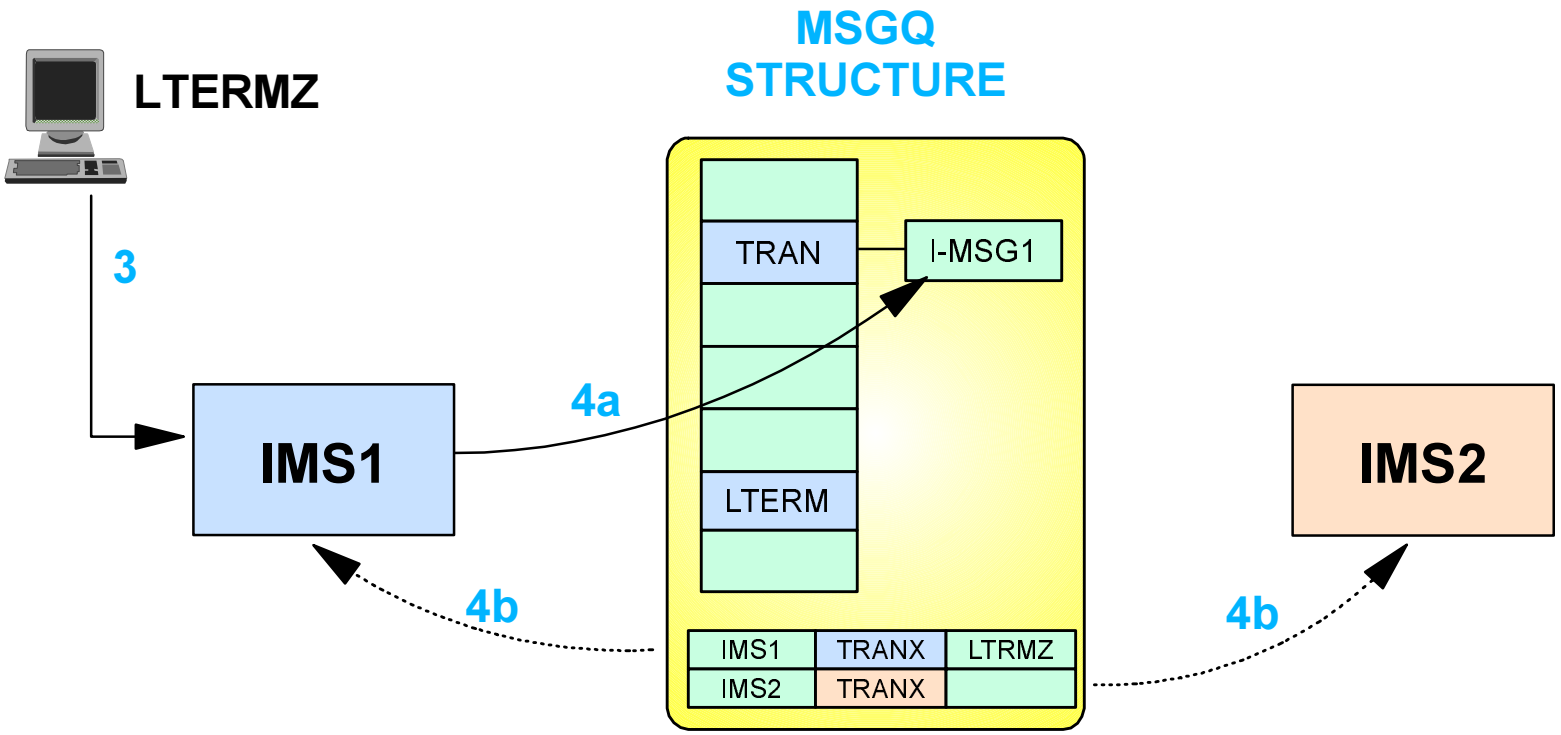
2. a) LTRMZ logs on to IMS1
   b) IMS1 registers interest in LTRMZ

IMS1 and IMS2

```
APPLCTN     PSB=PSBX,FPATH=NO
TRANSACT    CODE=TRANX

TERMINAL    NAME=NODE1
NAME        LTRMZ
```

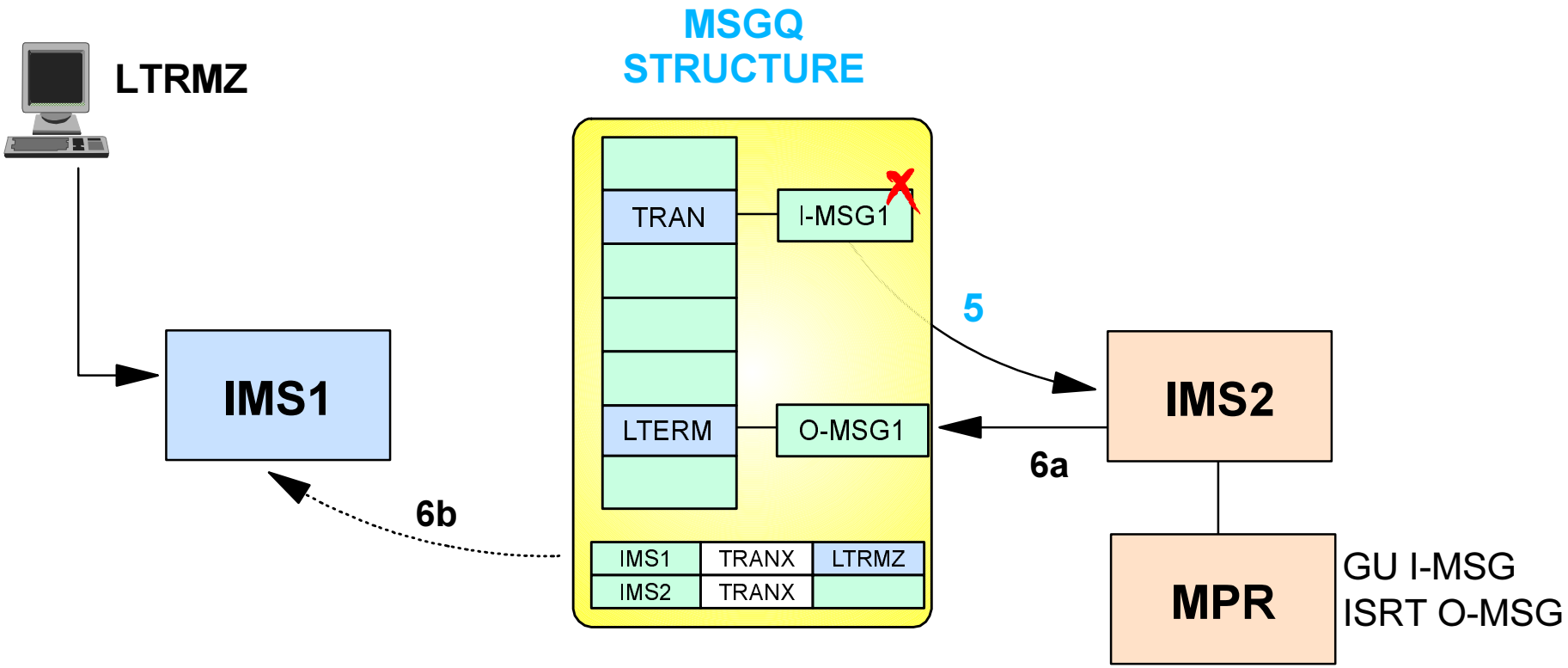
Shared Message Queue Example ...



3. LTERMZ sends I-MSG1 (TRANX) to IMS1
4. a) IMS1 places I-MSG1 on TRANX queue

   b) IMS1 and IMS2 are notified that there is work on the TRANX queue.

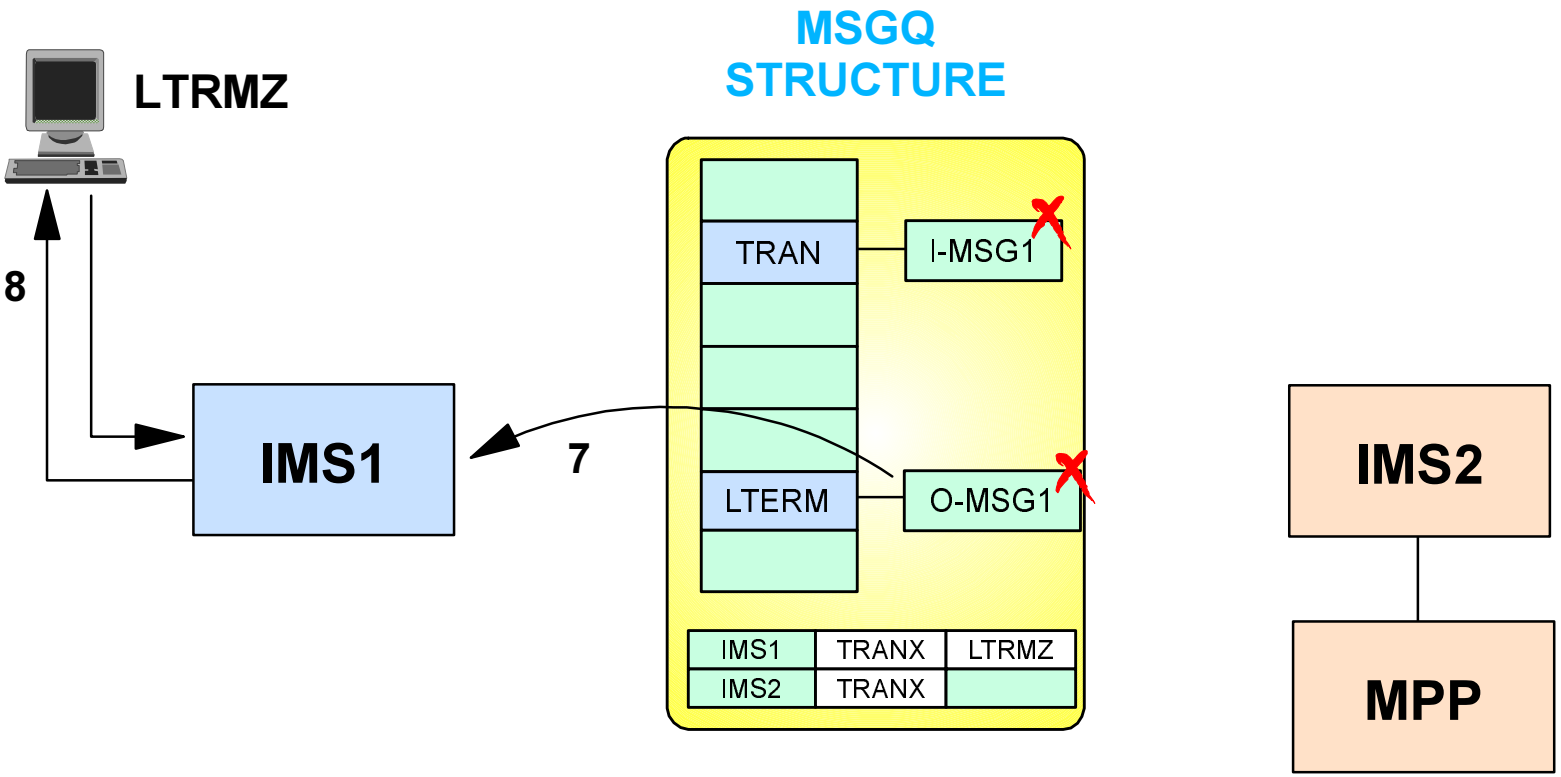
Shared Message Queue Example ...



5. IMS2 has an available MPR and retrieves I-MSG1 from TRANX queue

6. a) MPR processes I-MSG1 and IMS2 puts response (O-MSG1) on LTRMZ queue; I-MSG1 is deleted
   b) IMS1 is notified there is work on LTRMZ queue

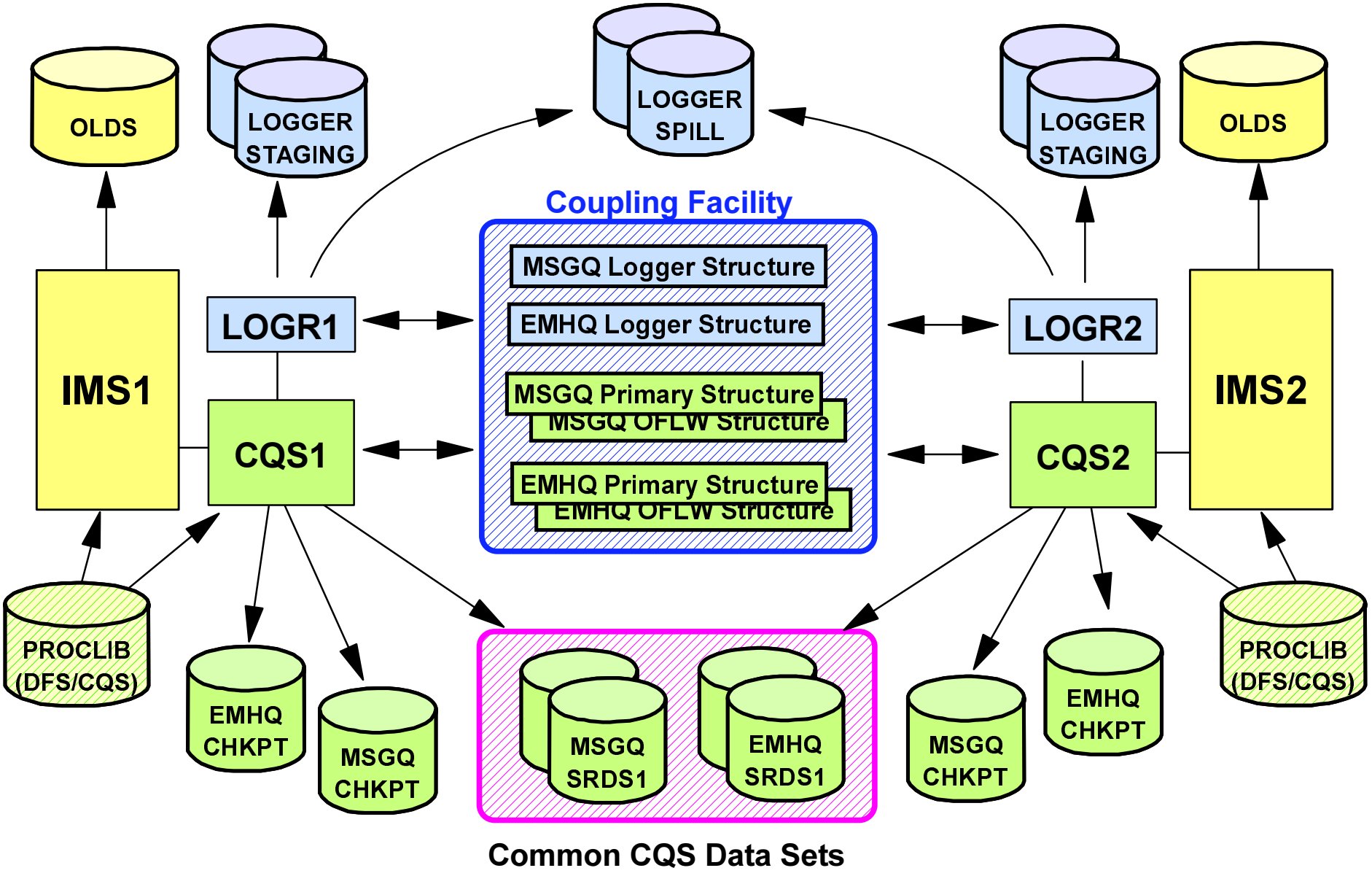
Shared Message Queue Example ...



7. IMS1 retrieves response (O-MSG1) from LTRMZ queue

8. IMS1 sends response to LTRMZ; O-MSG1 is deleted when LTRMZ ackknowledges.

Shared Queues Components



IMS, CQS, and SQ Interactions

Anaheim, California October 23 - 27, 2000

IMS Implementation of Shared Queues

Shared queues applies to both full function message queuing and fast path EMH message queuing

- Conventional Message Queues data sets and QPOOL queuing function
  - Replaced by **MSGQ structures** in the coupling facility
- Conventional EMH queuing function of the Balancing Groups (BALGs)
  - Replaced by **EMHQ structures** in the coupling facility

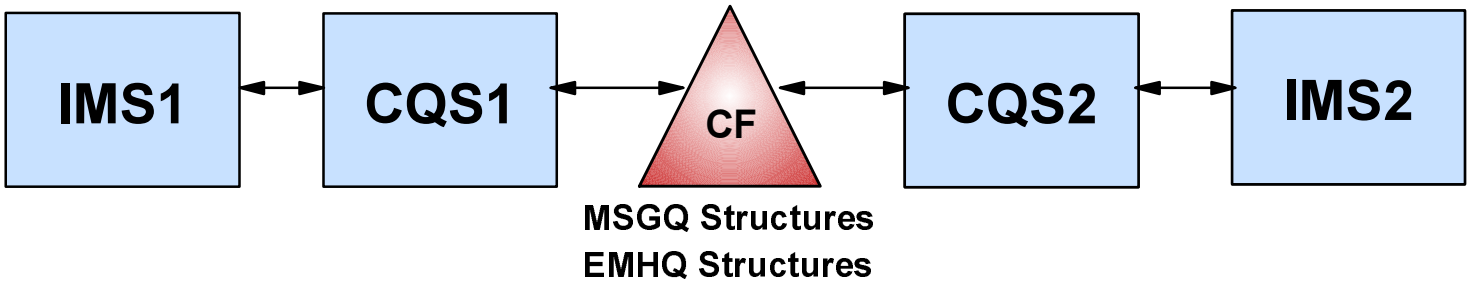
An IMS system is either fully Conventional Queues or fully Shared Queues

- Cannot use conventional queues for fast path and shared queues for full function
- Cannot use conventional queues for some transactions and shared queues for others

IMS Implementation of Shared Queues ...

IMS uses the Common Queue Server (CQS) to manage the shared queue structures on the coupling facility

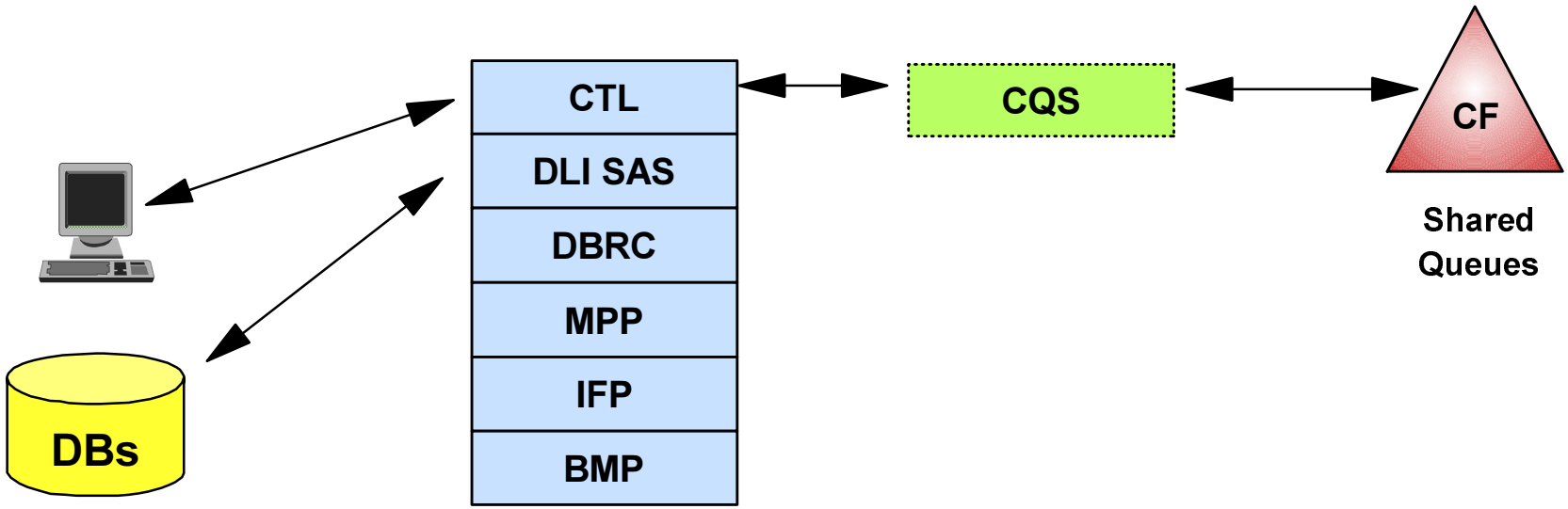
- Runs in separate address space connected to IMS
- Connects to shared queue structures
- Acts as **server** for IMS control region



IMS and CQS

IMS

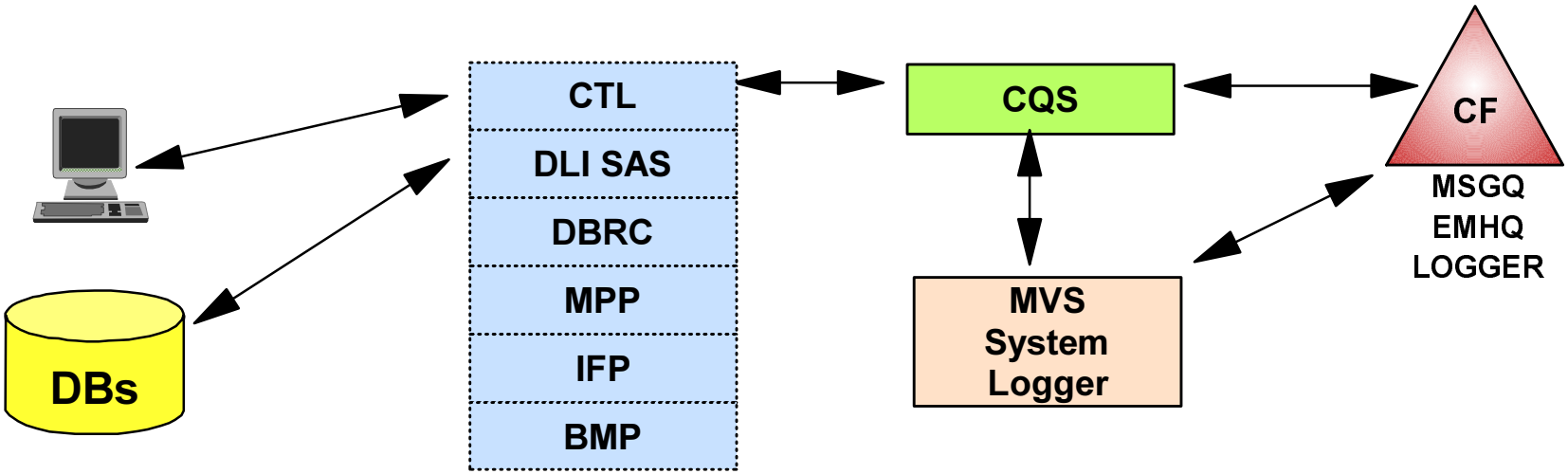
- Manages the network traffic
  - Sends/Receives messages
- Schedules and processes messages
  - Full function
  - Fast path
- Uses CQS services to manage the queue
  - Store and retrieve input and output messages
  - Provide message integrity



IMS and CQS ...

CQS

- Connects to the Shared Queue Structures
  - Builds structures if necessary
- Responsible for structure integrity
  - Structure checkpoint (image copy)
  - Logging (uses the MVS system logger)
  - Structure recovery (IC+ logs)
- Processes requests from IMS
  - Put, read, browse, move, delete, ...
- Informs IMS when there is work on the queues



Initialization

Start IMS with shared queues active

- **SHAREDQ=nnn**

Register with CQS

- IMS Starts CQS (if CQS not already active)
- CQS connects to shared queue structures

Connect to MSGQ and EMHQ structures through CQS

- CQS connects to MVS logstreams

Restart IMS

- /NRE or /ERE
- Resynchronize and recover (if necessary)
  - Resolve incomplete UOWs

Inform CQS of interest in specific queue names

- Register interest in transactions it can process (defined and not stopped)

Normal Operations

IMS

- Receives messages from multiple sources
  - Network - APPC
  - MSC - OTMA Clients
  - ISC - IMS Applications
- Issues requests to CQS to manage messages
  - PUT message on the queue
  - READ message from the queue
  - DELETE message from the queue
  - MOVE message to another queue
- Processes messages on the shared queue
  - INFORMs CQS of interest (or discontinued interest) in resources
  - Schedules transactions
  - Sends messages to their destinations

Normal Operations ...

CQS

- Provides services to IMS
  - Processes IMS requests
  - Informs IMS of work on the queues
  - Performs overflow processing
- Provides for message integrity
  - Logs activity to MVS logstreams
  - Performs system checkpoint processing
  - Performs structure checkpoint processing
  - Performs structure rebuild (if necessary)

MVS System Logger

- Processes CQS requests to log activity
  - Message traffic
  - CQS checkpoint data
  - Structure checkpoint data
- Manages the shared logstream
  - Multiple CQSs share a single logstream

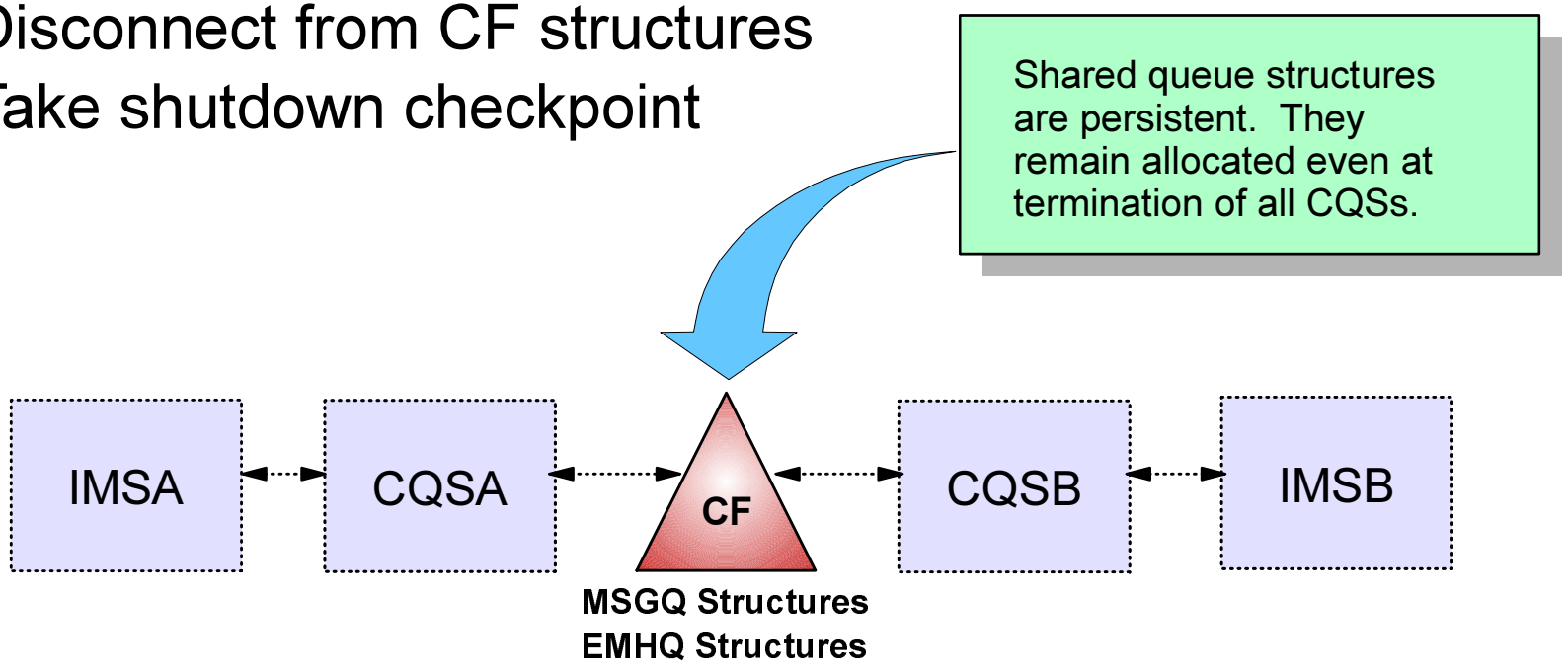
Termination

IMS Termination

- Disconnect from CQS
  - CQS deregisters interest in all queues
- Deregister from CQS

CQS termination

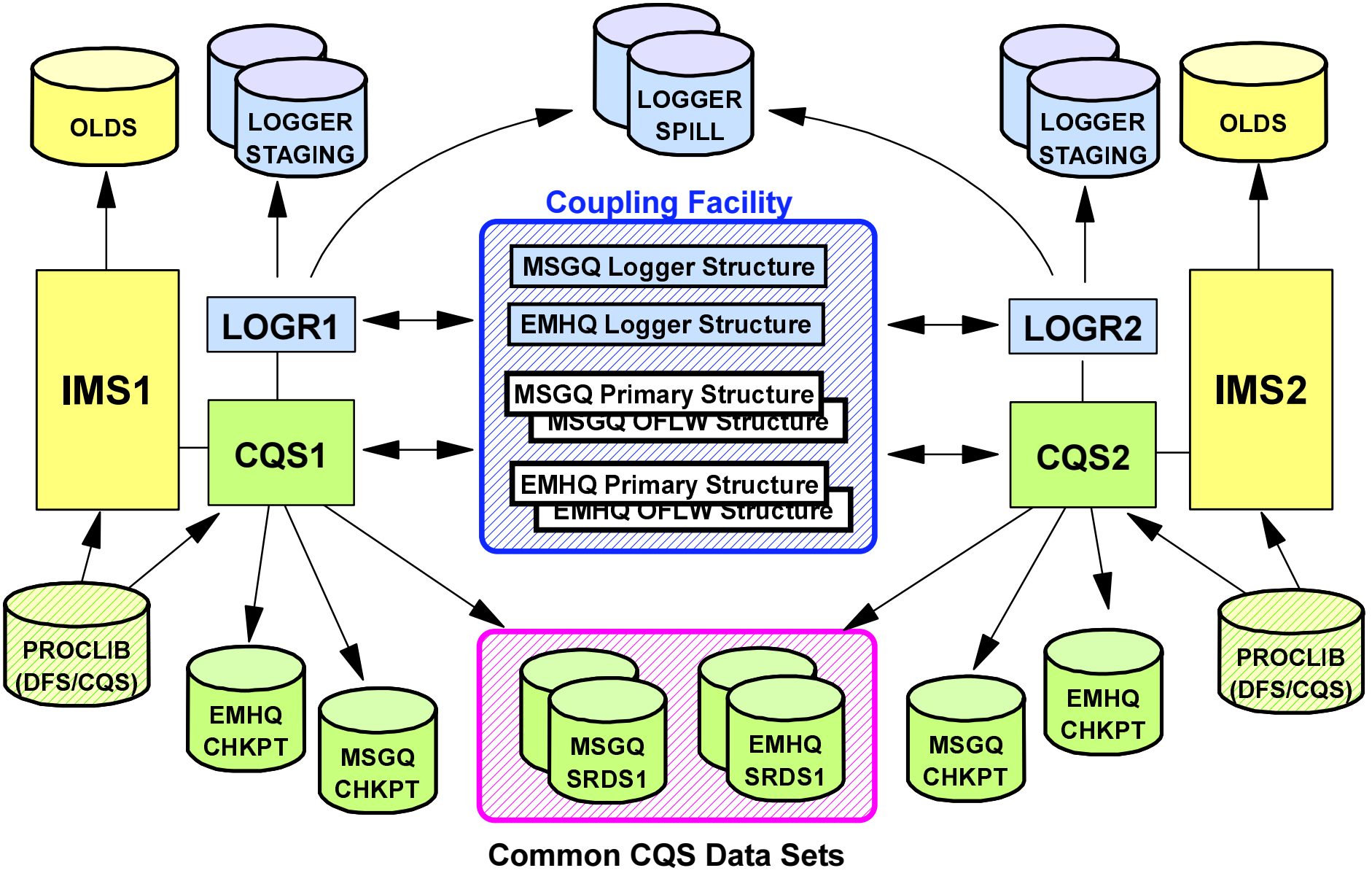
- Automatic when IMS deregisters (optional)
- Disconnect from logstreams
- Disconnect from CF structures
- Take shutdown checkpoint



Shared Queue Structures

Anaheim, California October 23 - 27, 2000

Shared Queues Components



Shared Queue Structures

Shared queue structures are defined in the CFRM Policy and allocated by CQS as list structures

- Full function message queue (**MSGQ**) structures
  - Primary
    - Contains full function input and output messages
  - Overflow (optional)
    - When primary structure reaches x-% full
- Fast path EMH message queue (**EMHQ**) structures
  - Primary (required only if fast path is enabled - FPCTRL macro)
    - Contains EMH input and output messages
  - Overflow (optional)
    - When primary structure x-% full

Shared Queue Structures ...

CFRM Policy contains definitions for each of these structures

- First CQS to connect **allocates** structure in the Coupling Facility
- Structures are persistent
  - They remain allocated even if all users (CQS) have disconnected
  - They remain allocated even if all IMSs and their associated CQSs cold start
- Structures are recoverable
  - Coupling facility failure
  - Structure failure
  - Connection failure

CFRM Policy may be STARTed (activated) on any MVS

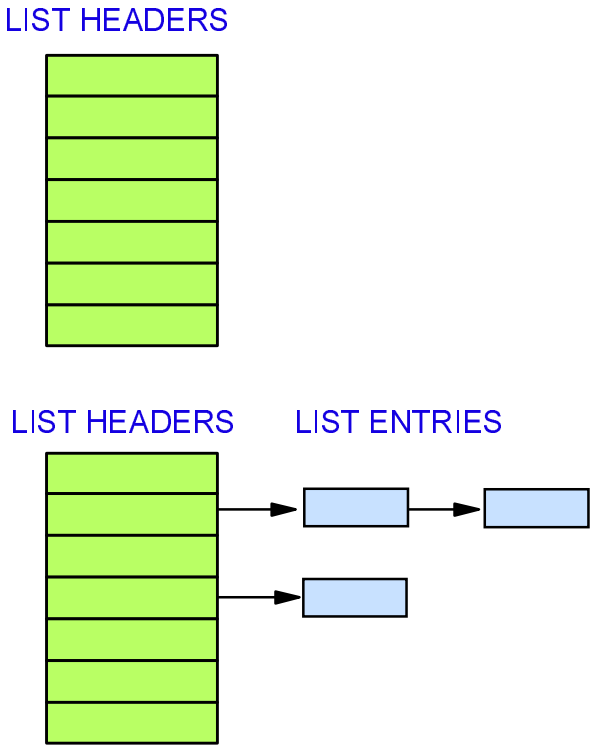
- Effective for all MVSs in Sysplex

```
SETXCF  START,POLICY,TYPE=CFRM,POLNAME=policyname
```

List Structure Components

Two list structure components are siginificant

- List headers
  - Anchor for list entries
  - 256 available per structure
  - CQS uses 192 list headers
- List entries
  - List entry controls
  - Messages (stored in one or more data elements of 512 bytes each)
  - Large messages may require multiple list entries



IMS/CQS Use of List Structures

Anaheim, California October 23 - 27, 2000

IMS/CQS Use of List Headers

192 list headers are assigned at structure allocation time

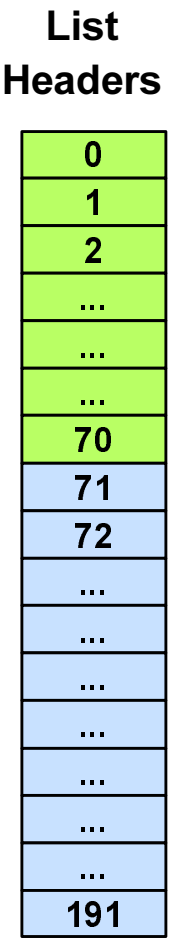
- List headers are anchor points for CQS or IMS queues

CQS uses (or reserves) the first 71 list headers

- LH 0 - 70
- Referred to as **Private Queues**
- Used to control or manage the list structures
- CQS determines meaning of each list header
  - e.g., LH 0 is Control Queue

IMS uses (or reserves) the last 121 list headers

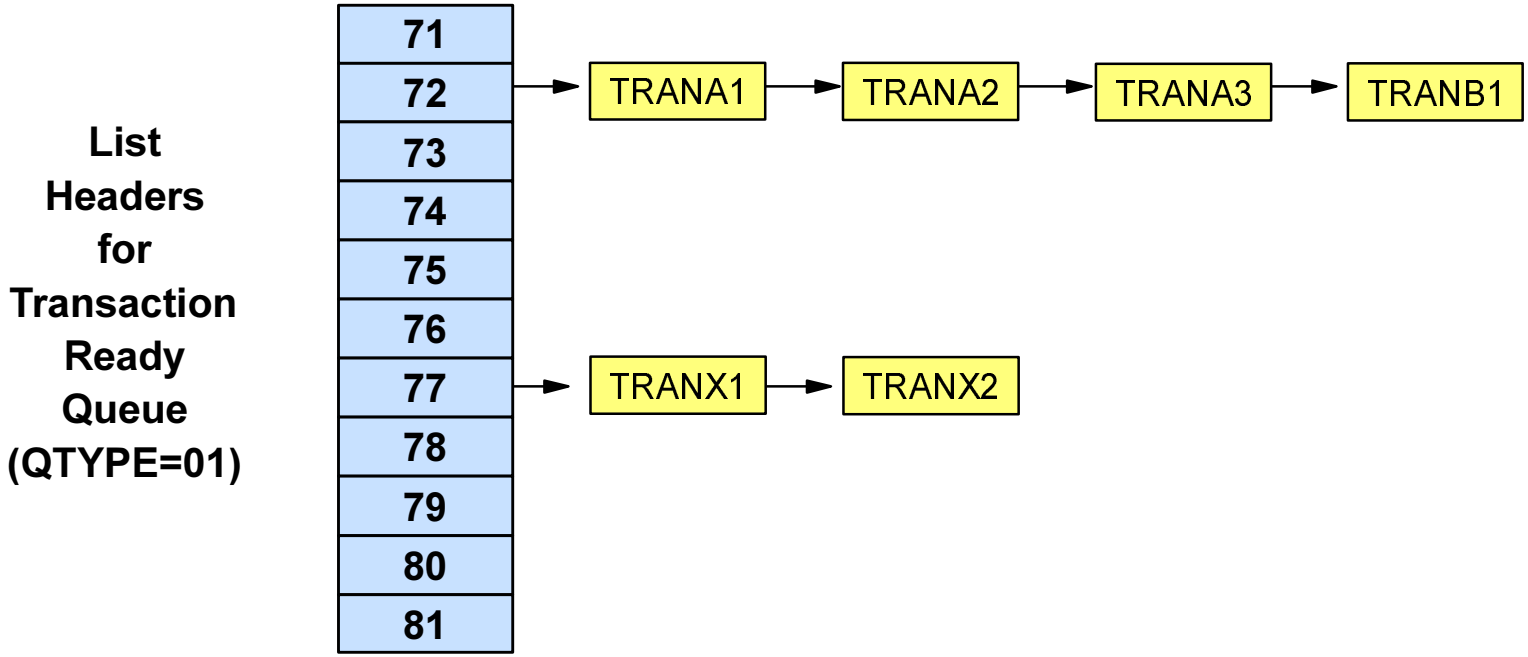
- LH 71 - 191
- Referred to as **Client Queues**
- Used to anchor lists (queues) of IMS messages
- IMS determines meaning of each list header
  - e.g., LH 71-81 are Transaction Queues



IMS Client Queues (Full Function)

IMS uses its 121 list headers (71-191) for client queues

- IMS currently defines **9 queue types**
- These 9 queue types use **99 list headers**
  - Multiple (11) list headers per queue type are used to reduce contention for access to the queue (list)
- The other 22 list headers are not currently used by IMS



IMS Client Queue Types (Full Function)

1. **Transaction Ready Queue (LH 71-81)**
   - Messages destined for **local or remote transactions**
2. **Transaction Staging Queue (LH 82-92)**
   - Non-first portion of **long transaction messages**
     - i.e., Message requires multiple queue buffer logical records
3. **Transaction Suspend Queue (LH 93-103)**
   - Transaction messages which have been **suspended** (e.g. 3303 abend)
4. **Transaction Serial Queue (LH 104-114)**
   - Messages destined for transactions defined as **SERIAL=YES**
5. **LTERM Ready Queue (LH 115-125)**
   - Messages destined for **local LTERMs**

IMS Client Queue Types (Full Function) ...

6. **LTERM Staging Queue (LH 126-136)**
   - Non-first portion of **long output messages** destined for local LTERMs, remote LTERMs, MSNAMEs, OTMA clients, or APPC devices
7. **APPC Ready Queue (LH 137-147)**
   - **Output messages** destined for APPC devices
   - Input messages queued on the Trans. Ready Queue
8. **Remote Ready Queue (LH 148-158)**
   - Messages destined for **MSNAMES or remote LTERMs**
9. **OTMA Ready Queue (LH 159-169)**
   - **Output messages** destined for OTMA devices
   - Input messages queued on the Trans. Ready Queue

Two additional queue types are reserved (LH 170-191)

IMS Client Queue Types (Fast Path)

Fast path has its own list structure(s)

- EMHQ structure

1. **Program Ready Queue (LH 71-81)**
   - **Input messages** destined for EMH (IFP) programs
2. **LTERM Ready Queue (LH 115-125)**
   - **Responses** to EMH input

EMHQ uses only two queue types

- Staging queues are never required
  - Input and output messages must fit in one EMH buffer
- APPC and OTMA responses are not put on shared queues
  - Delivered directly from EMH buffers
- Serial and suspended transactions are not supported by EMH
- MSC output (ISRT to ALT-PCB) uses full function MSGQ structure

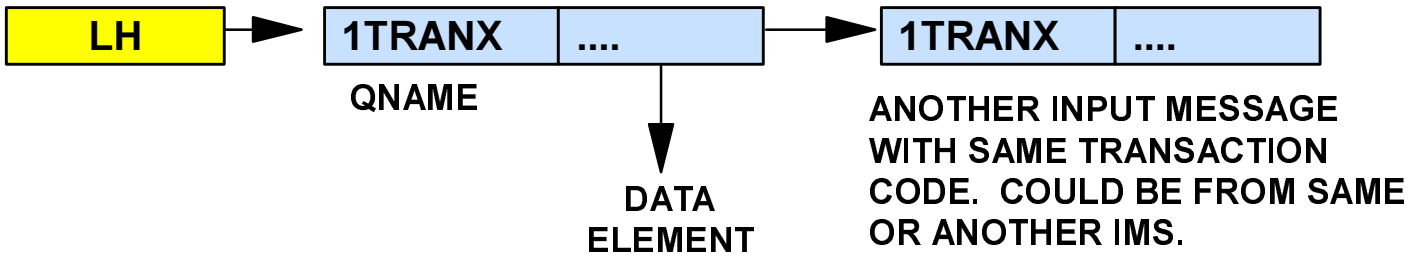
IMS Queue Names

Queue names are used by IMS/CQS to register interest

- Transactions, LTERMs, MSNAMEs
- When queues go from empty to non-empty, IMS/CQS is informed

List Entry Controls

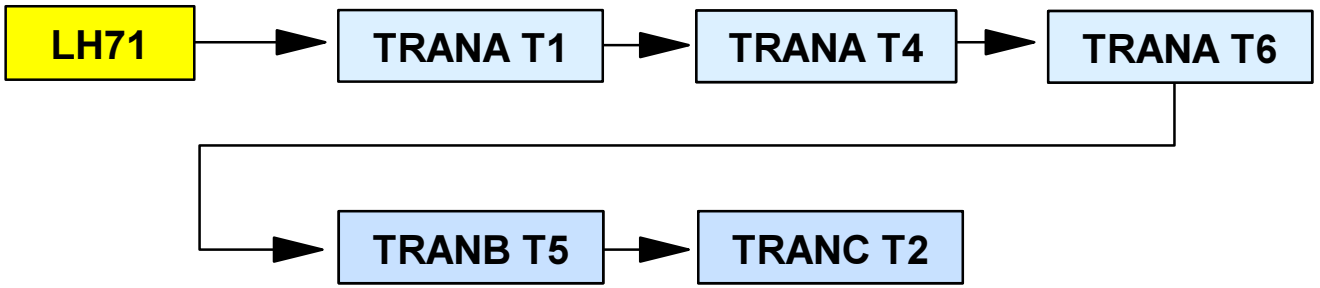
- Entry Key (16 bytes) includes **Queue Name**
  - May be non-unique (synonyms are in FIFO sequence
- Queue name
  - First byte of queue name is always **queue type**
  - Remainder of queue name depends on queue type and message
    - Usually includes IMS destination name



Queuing Messages on a List Header

Each Qtype uses 11 list headers

- Each list header must support multiple queue names
  - e.g. Multiple transaction codes on Qtype 01
- Multiple queue names are queued in collating sequence
  - e.g. TRANA-TRANB-TRANC-.....
- Two messages with the same queue name are synonyms
  - Messages with same queue name form a **sublist**
  - Messages are queued FIFO within a sublist



Long Messages

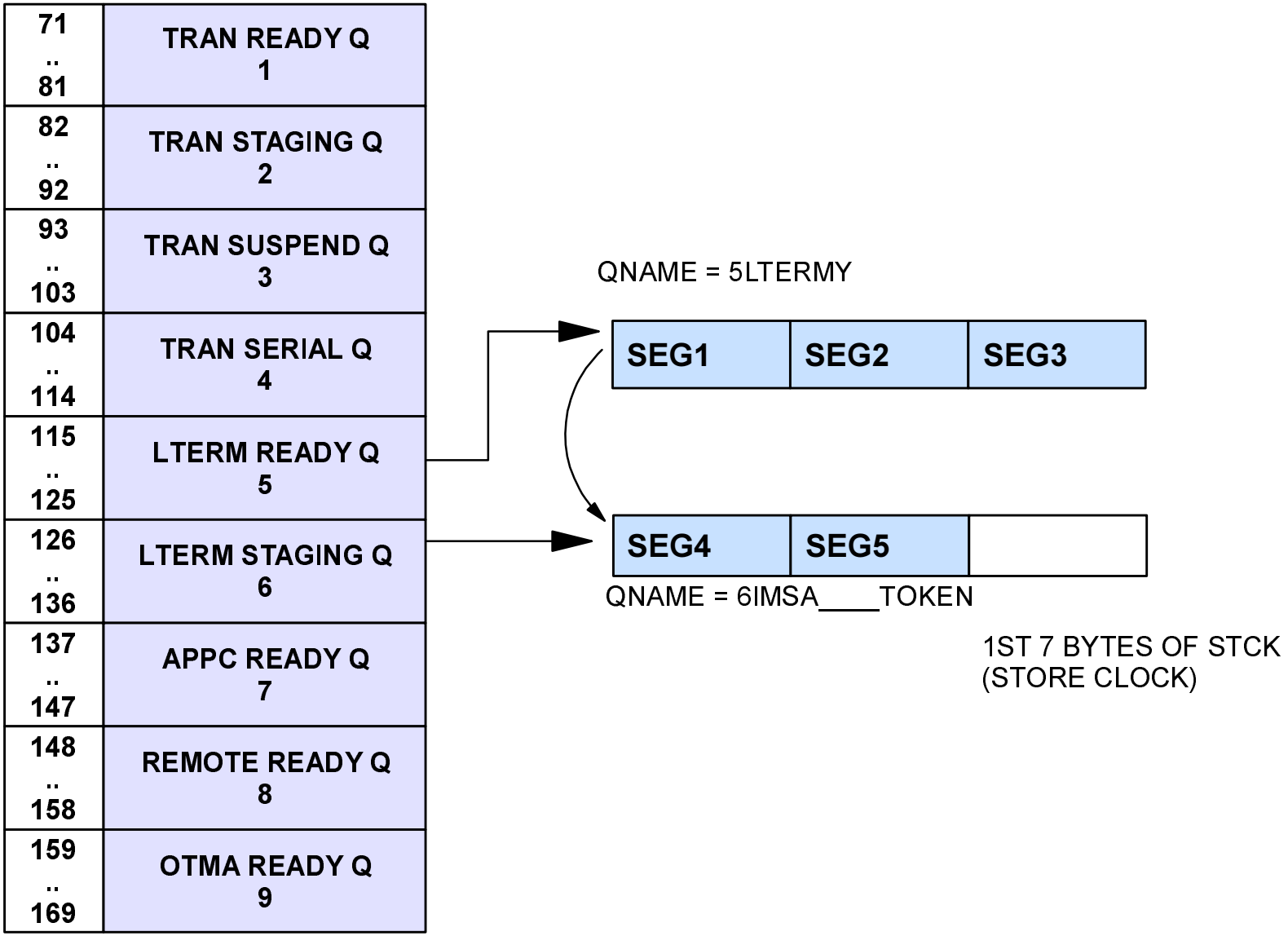
If a message requires more than one QBuffer logical record

- 2nd - nth logical records are placed on Staging Queue
- 1st logical record is placed on Ready Queue
  - 1st logical record is always put on structure last (after commit)
  - Eliminates chance another registered IMS will try to read 1st QBuffer before other logical records are placed on staging queue
- IMS Message Prefix contains Queue Name of list entry on Staging Queue

Example

- An application in IMSA inserted one output message to LTERMY containing 5 segments requiring 2 QBuffer logical records
- **First** ... the contents of Logical Record 2 (SEG4, SEG5) are placed on the LTERM Staging Queue
- **Then** ... the contents of Logical Record 1 (SEG1, SEG2, SEG3) are placed on the LTERM Ready Queue

Long Messages ...



CQS Use of List Headers

CQS has 71 list headers for its private queues

- Used to control or manage the list structures

Control Queue
Cold Queue
Lock Queue
Rebuild Queue
Move Queue
Delete Queue

Intermediate queues used to maintain message integrity for certain structure operations.

Registering Interest in Specific Queues

Registration

- IMS **registers interest in a specific queue name**
  - At IMS initialization for statically defined transactions
  - When status changes
    - LU for static LTERM(s) logs on
    - LU for dynamic (ETO) LTERM signs on
    - MSC logical link started
    - Stopped transaction has been started
    - Transaction has been added by online change

Deregistration

- IMS **deregisters interest in specific queue name**
  - At IMS termination (actually CQS does it for IMS)
  - When IMS is no longer capable of processing the queue
    - An LTERM is stopped
    - Static LTERM logs off
    - Dynamic LTERM signs off
    - Transaction is stopped
    - MSC logical link is stopped
    - Transaction is deleted by online change

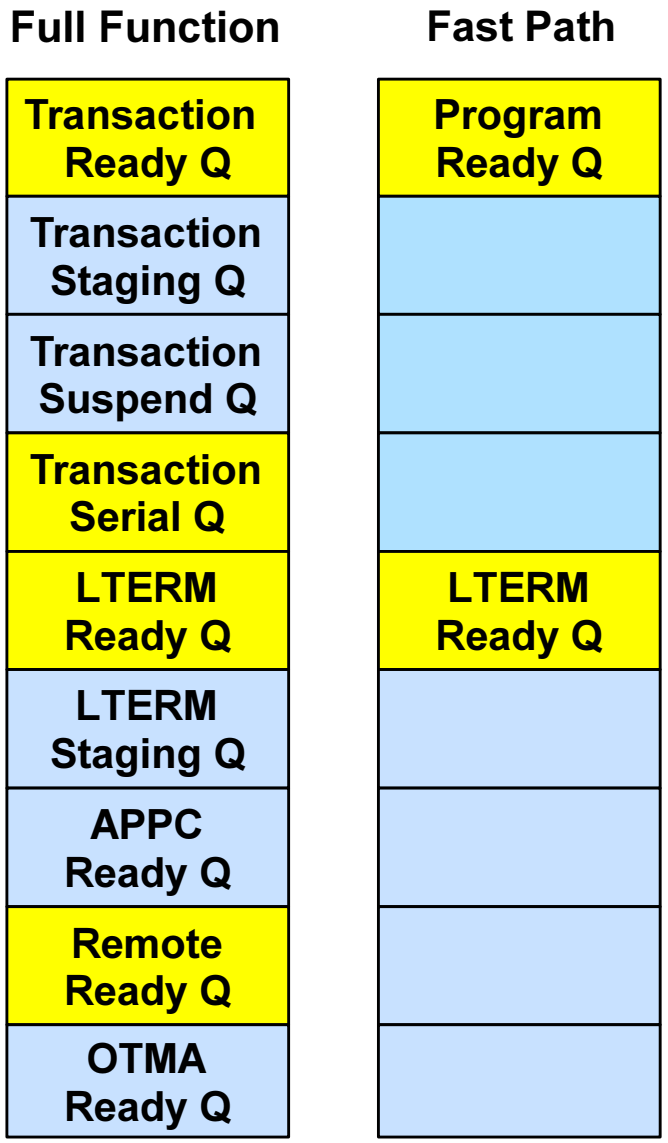
Queue Types

Transaction Ready
Transaction Staging
Transaction Suspend
Transaction Serial
LTERM Ready
LTERM Staging
APPC Ready
Remote Ready
OTMA Ready

Registering Interest in Specific Queues ...

IMS registers interest in a specific queue name on a specific queue

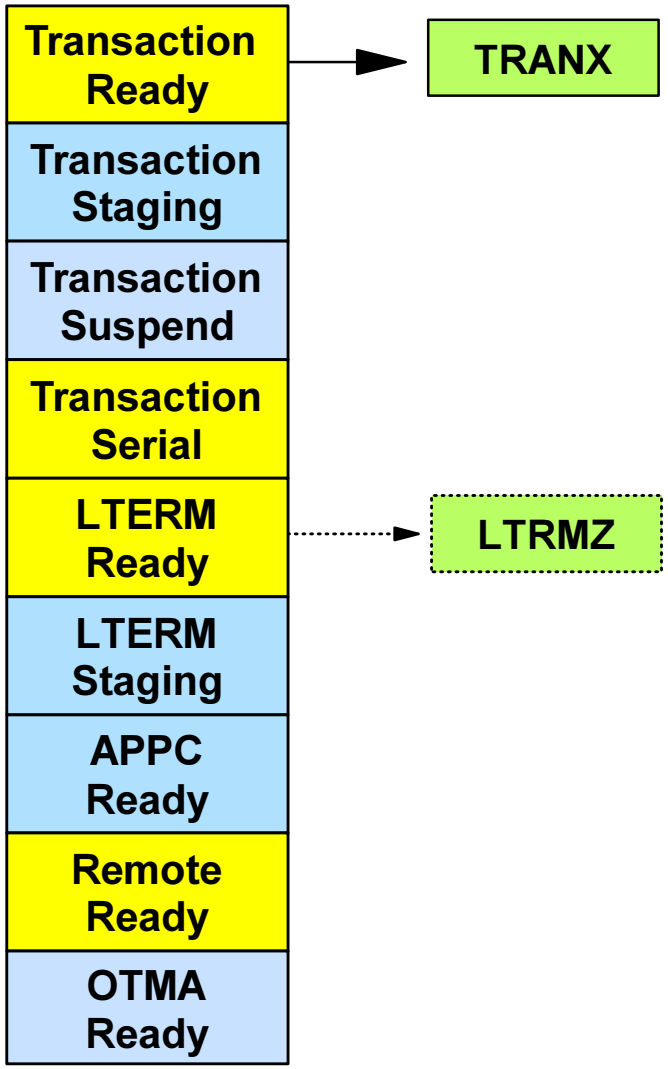
- **If it is capable of processing a message on that queue**
  - Wants to be informed when a message is placed on that queue
- **IMS does not register interest in**
  - Staging Queues
  - Suspend Queue
  - APPC Ready Queue
  - OTMA Ready Queue



Registering Interest in Specific Queues ...

IMS is informed of work on a registered queue name ...

- If there is **work on the queue when IMS registers interest**
- When the queue goes from **empty to non-empty**
- When a queue goes from **non-empty to empty**
  - IMS is not informed



Accessing the Shared Queues

Anaheim, California October 23 - 27, 2000

IMS Access to the Shared Queues

IMS interface to CQS for full function services is through the IMS Queue Manager

- New queue manager for Shared Queues environment
- IMS QPOOL is used for intermediate storage of messages
  - Before they are put on the shared queues
  - When they have been retrieved for processing
  - To save the SPA and last output message for conversations
  - When there is a problem with CQS or the structure
- Messages on the structures are the basis for all message queue integrity and recovery

IMS interface to CQS for fast path services is through the Expedited Message Handler (EMH)

- New options in DBFHAGU0

Eligibility for Sysplex-wide Processing

Most messages are placed on the shared queue structures

- Some exceptions
  - Commands
  - Command responses not destined for secondary master terminal
  - Fast path input and output messages designated by DBFHAGU0 for local processing
  - All fast path APPC/OTMA output

Most messages placed on the shared queue structures are eligible for Sysplex-wide processing

- Can be processed by any IMS **with registered interest** in the queue name (transaction, logical terminal, or MSNAME)
- Some exceptions
  - SERIAL transactions
  - APPC and OTMA input and output messages
  - Command responses destined for the secondary master terminal

Transaction Scheduling (FF)

When dependent region becomes available

- IMS performs all scheduling functions except passing control to dependent region controller
  - Pseudo-scheduling (allocates a region, loads the PSB, ...)
- IMS issues READ request to CQS
  - Gets first list entry on named queue sublist
    - Occupies one QBuffer logical record
  - CQS moves message from Ready Queue to Lock Queue
    - Prevents transaction from being retrieved more than once
- If long message, IMS issues BROWSE request to CQS
  - Retrieves message continuations from Staging Queue
- When entire message has been retrieved into QBuffers
  - Dependent region is given control
  - Issues GU/GN to IO-PCB to retrieve message

Transaction Scheduling (FF) ...

If ...

- Dependent region is available when input transaction arrives,
- Input message fits within a single QBuffer logical record, and
- Transaction not defined as SERIAL

Then ...

- Message is placed on LOCKQ instead of Ready Queue
- Message is retained in QBuffers
- Region is scheduled and control is passed to region controller
- Reduces overhead of
  - Putting message on Ready Queue and then reading it again and moving it to LOCKQ
  - Informing other IMSs that queue is not empty, saving (perhaps) pseudo-scheduling failures in other IMSs

To increase probability of this

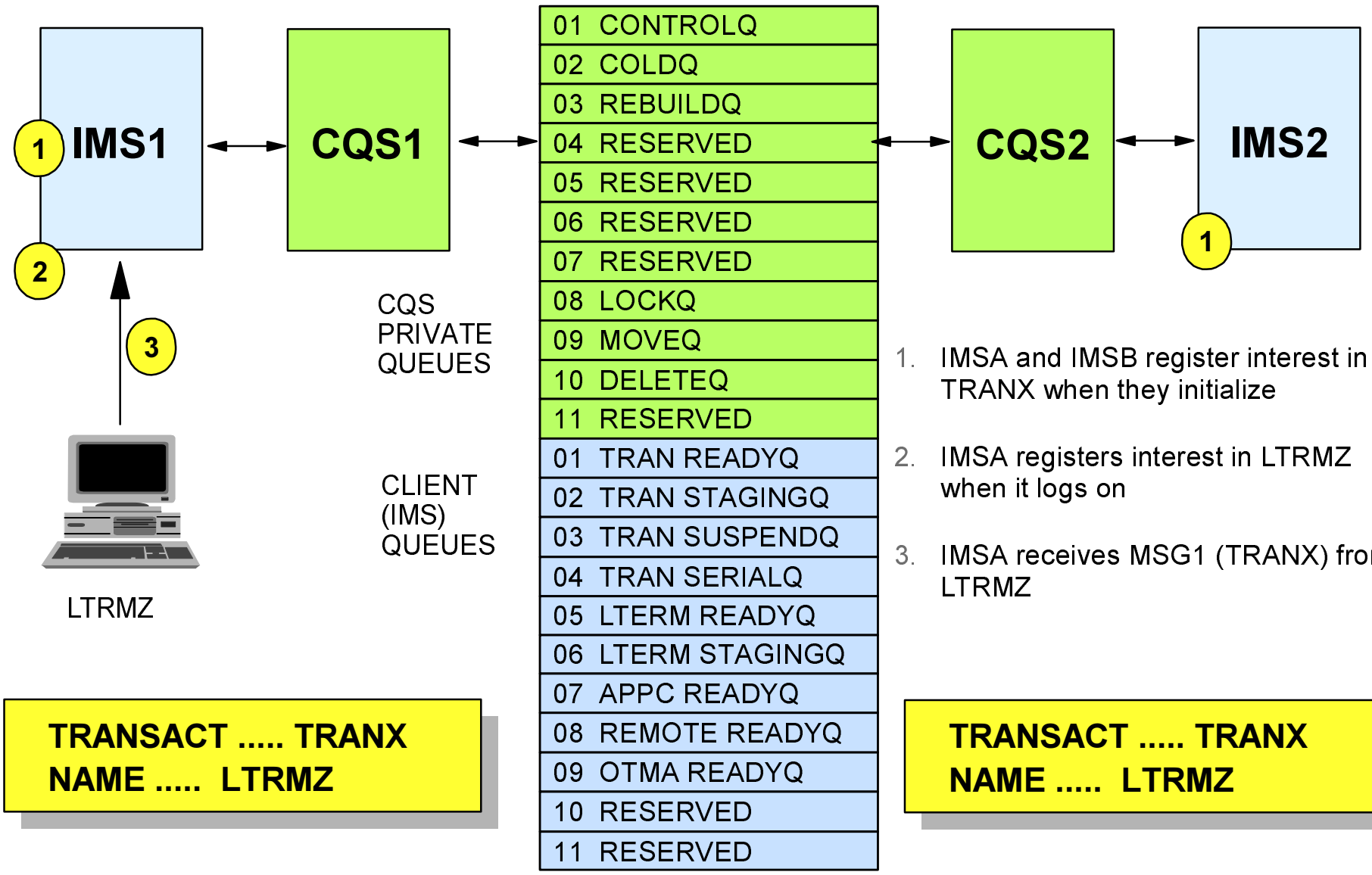
- Consider more dependent regions and fewer classes

Transaction Processing (FF)

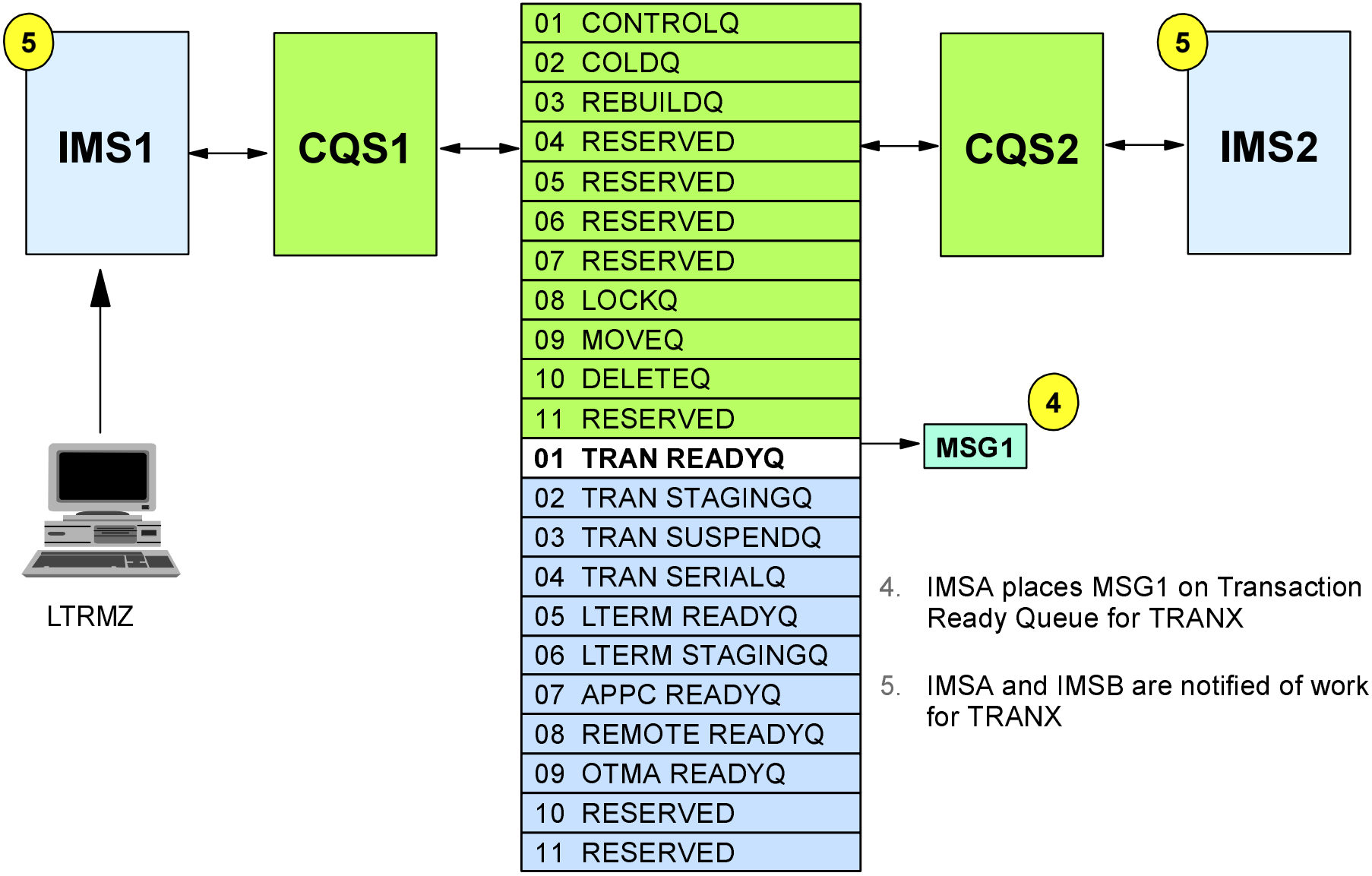
Application program output (ISRT) messages are placed (PUT) on the shared queue according to their destination

- At ISRT time
  - Message segments are stored in QBuffers
- When application commits
  - Input message is deleted
  - Output messages are queued to their final destination
    - LTERM Ready Queue
    - APPC/OTMA Ready Queue
    - Remote Ready Queue (Remote LTERM or MSNAME)
    - Transaction Ready Queue (Pgm-Pgm switch)
    - Transaction Serial Queue (Pgm-Pgm Switch)

Example FF Transaction Flow



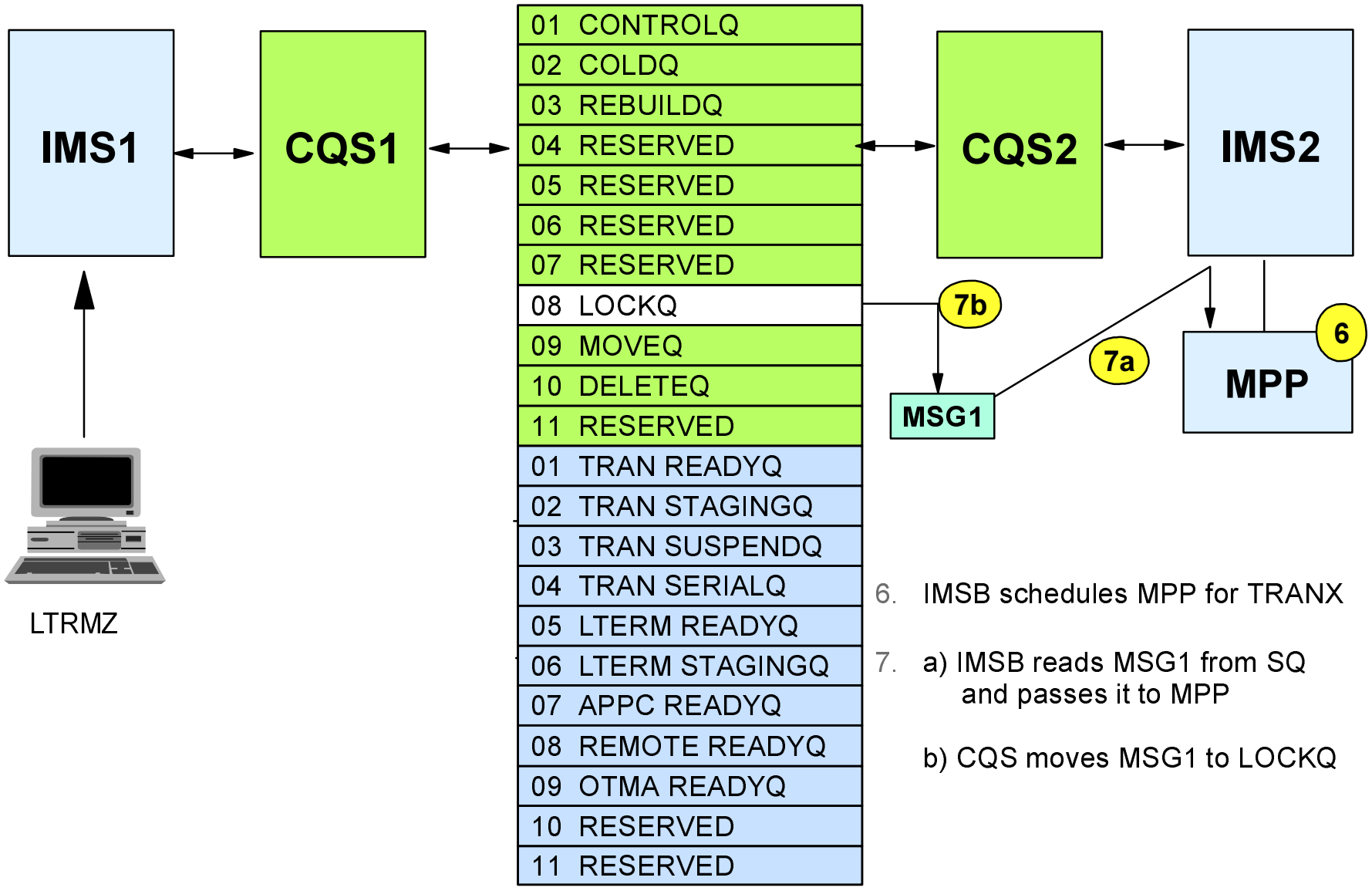
Example FF Transaction Flow ...



4. IMSA places MSG1 on Transaction Ready Queue for TRANX

5. IMSA and IMSB are notified of work for TRANX

Example FF Transaction Flow ...

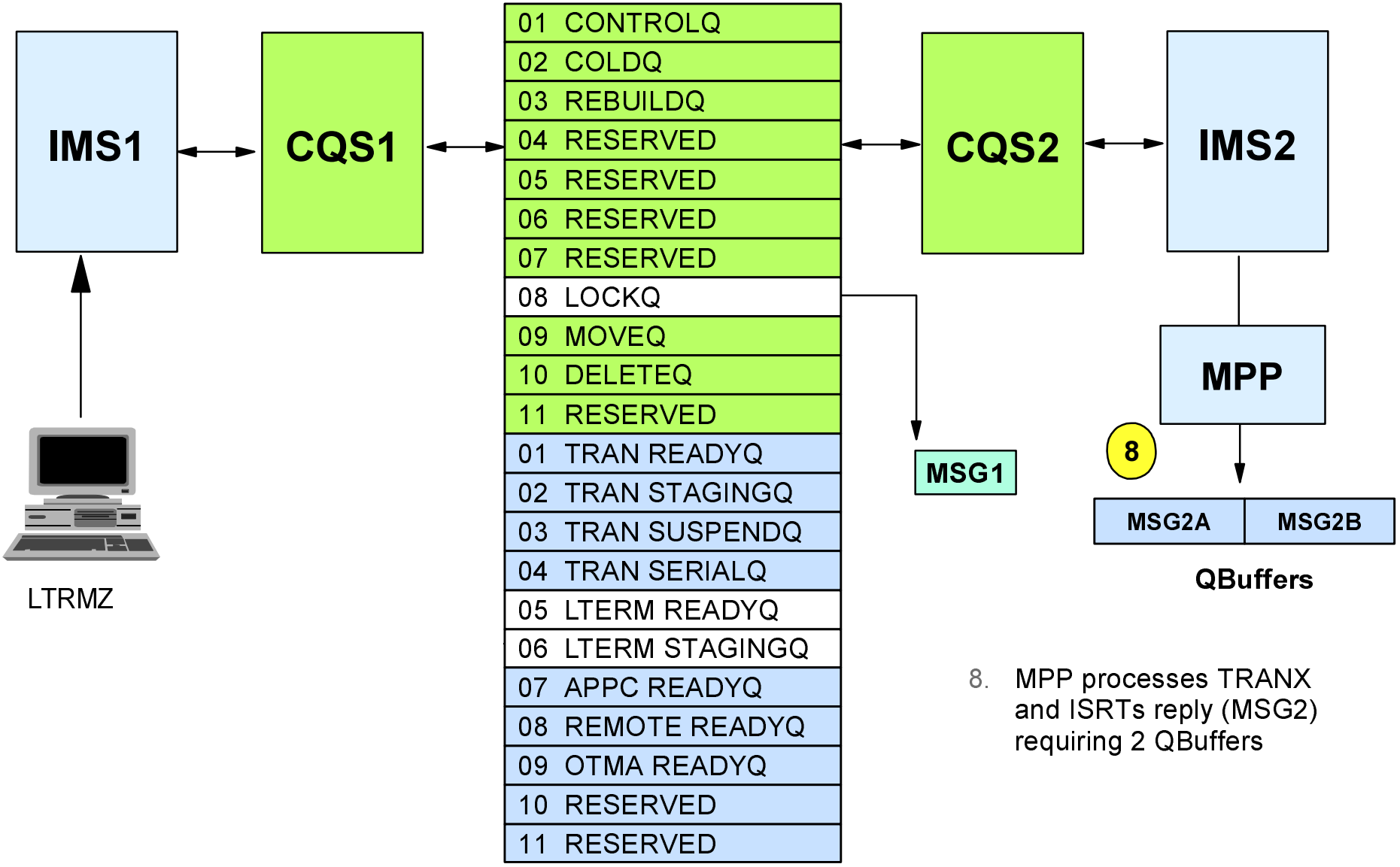


6. IMSB schedules MPP for TRANX

7. a) IMSB reads MSG1 from SQ and passes it to MPP

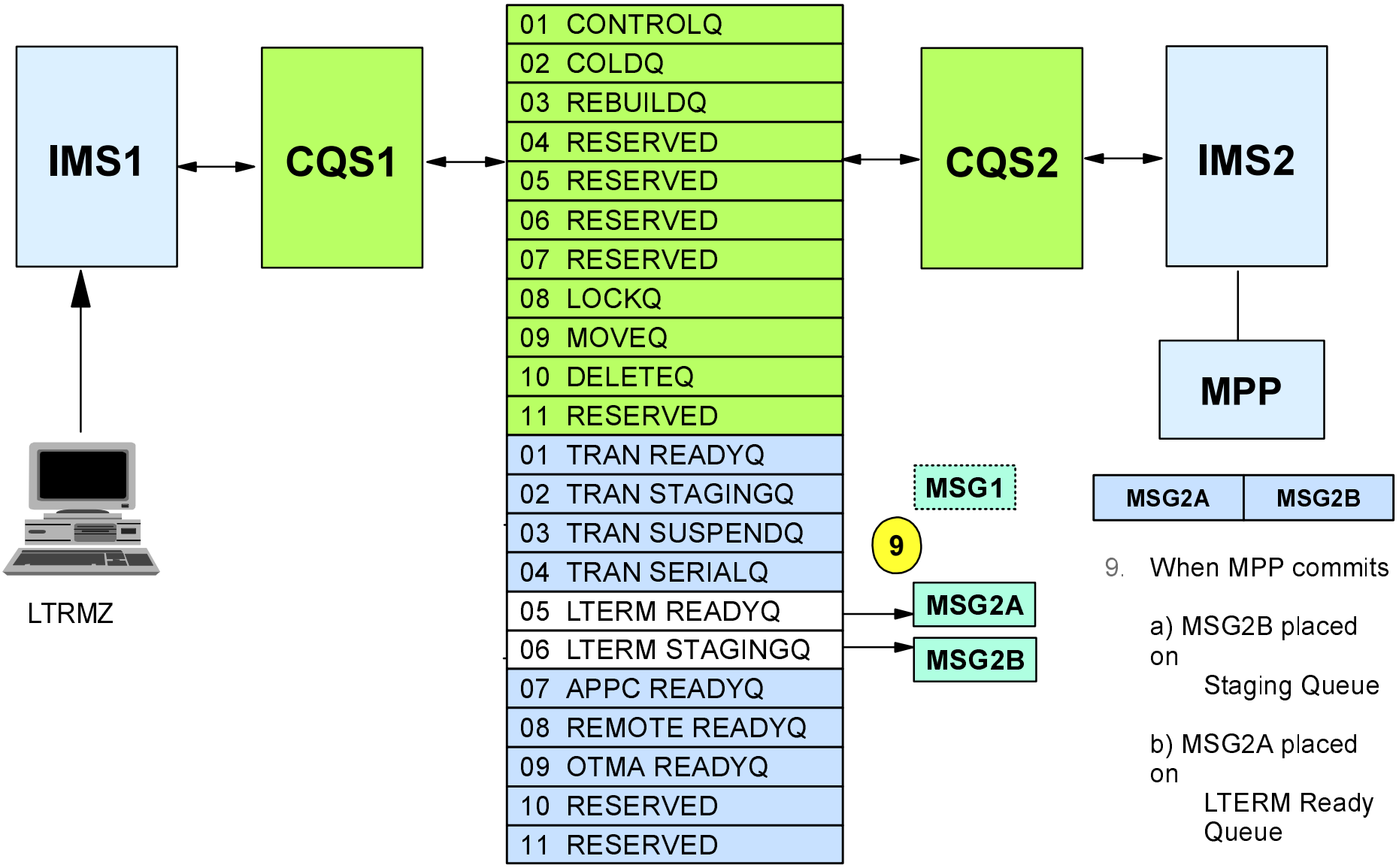
   b) CQS moves MSG1 to LOCKQ

Example FF Transaction Flow ...



8. MPP processes TRANX and ISRTs reply (MSG2) requiring 2 QBuffers

Example FF Transaction Flow ...



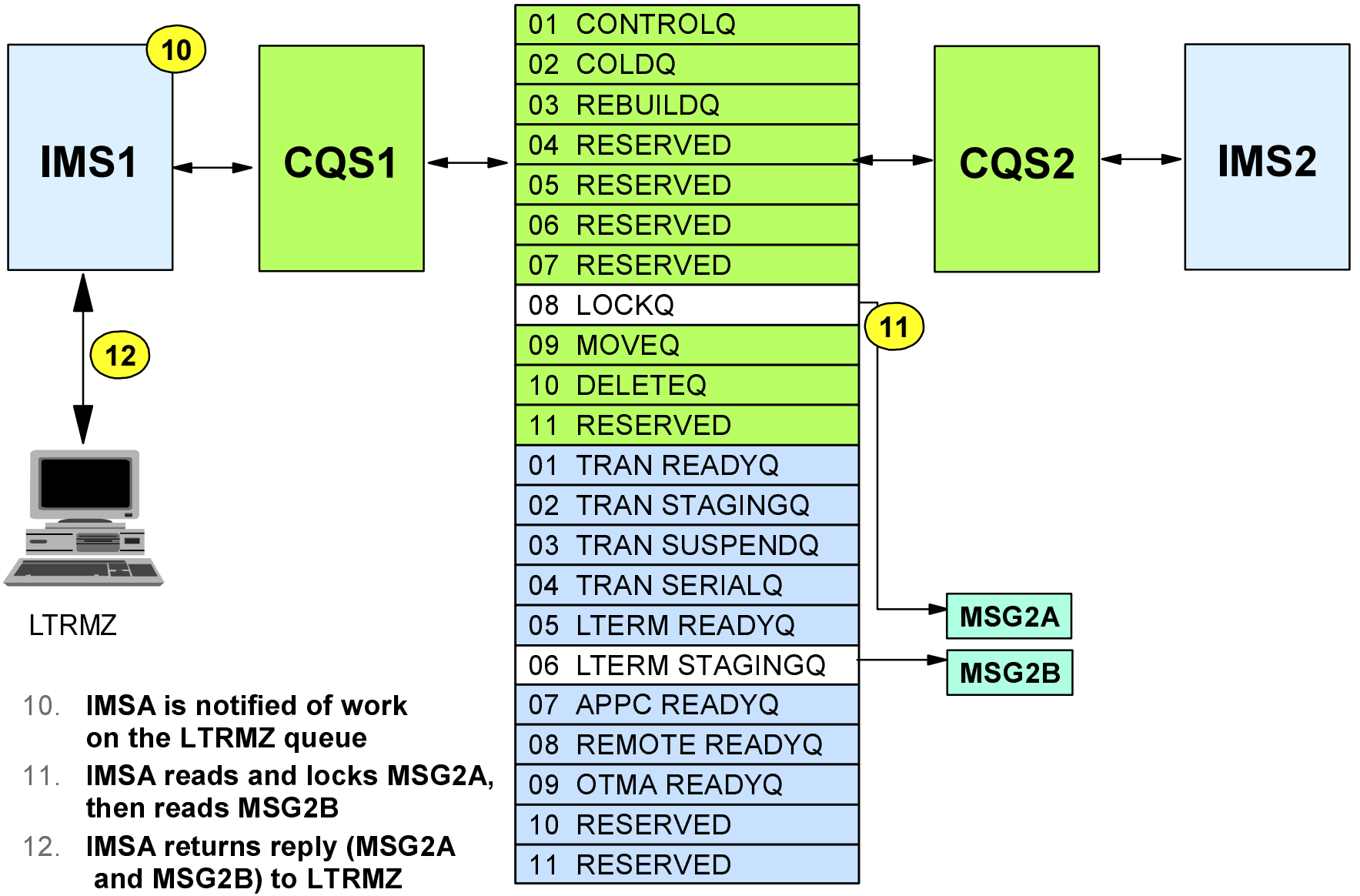
9. When MPP commits

   a) MSG2B placed on Staging Queue

   b) MSG2A placed on LTERM Ready Queue

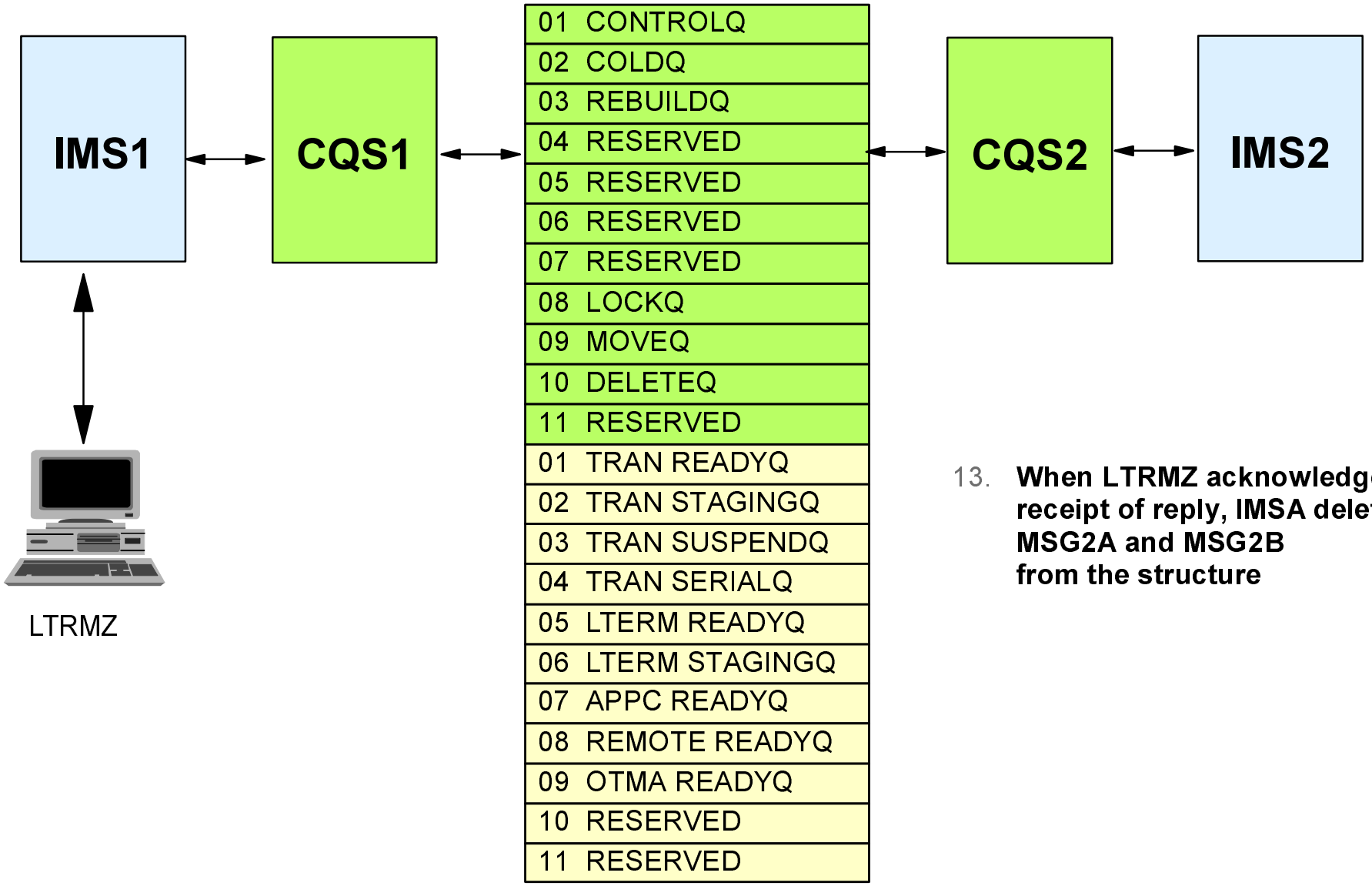
   c) MSG1 is deleted from LOCKQ

Example FF Transaction Flow ...



10. **IMSA is notified of work on the LTRMZ queue**
11. **IMSA reads and locks MSG2A, then reads MSG2B**
12. **IMSA returns reply (MSG2A and MSG2B) to LTRMZ**

Example FF Transaction Flow ...



13. **When LTRMZ acknowledges receipt of reply, IMSA deletes MSG2A and MSG2B from the structure**

Transaction Scheduling (EMH)

If input message is fast path potential or fast path exclusive transaction

- Message is passed to Fast Path Input Edit/Routing Exit (**DBFHAGU0**)
  - New **Extended Parameter List** passed to exit
- DBFHAGU0 knows which fast path PSBs are active in the shared queues group
  - **Local PSB Name Table**
  - **Global PSB Name Table**
- DBFHAGU0 has new scheduling options
  - **Sysplex Processing Code**
    - Determines whether transaction is to be scheduled locally or globally
  - **Schedule by PSB Name**
    - Overrides Routing Code scheduling

Sysplex Processing Code

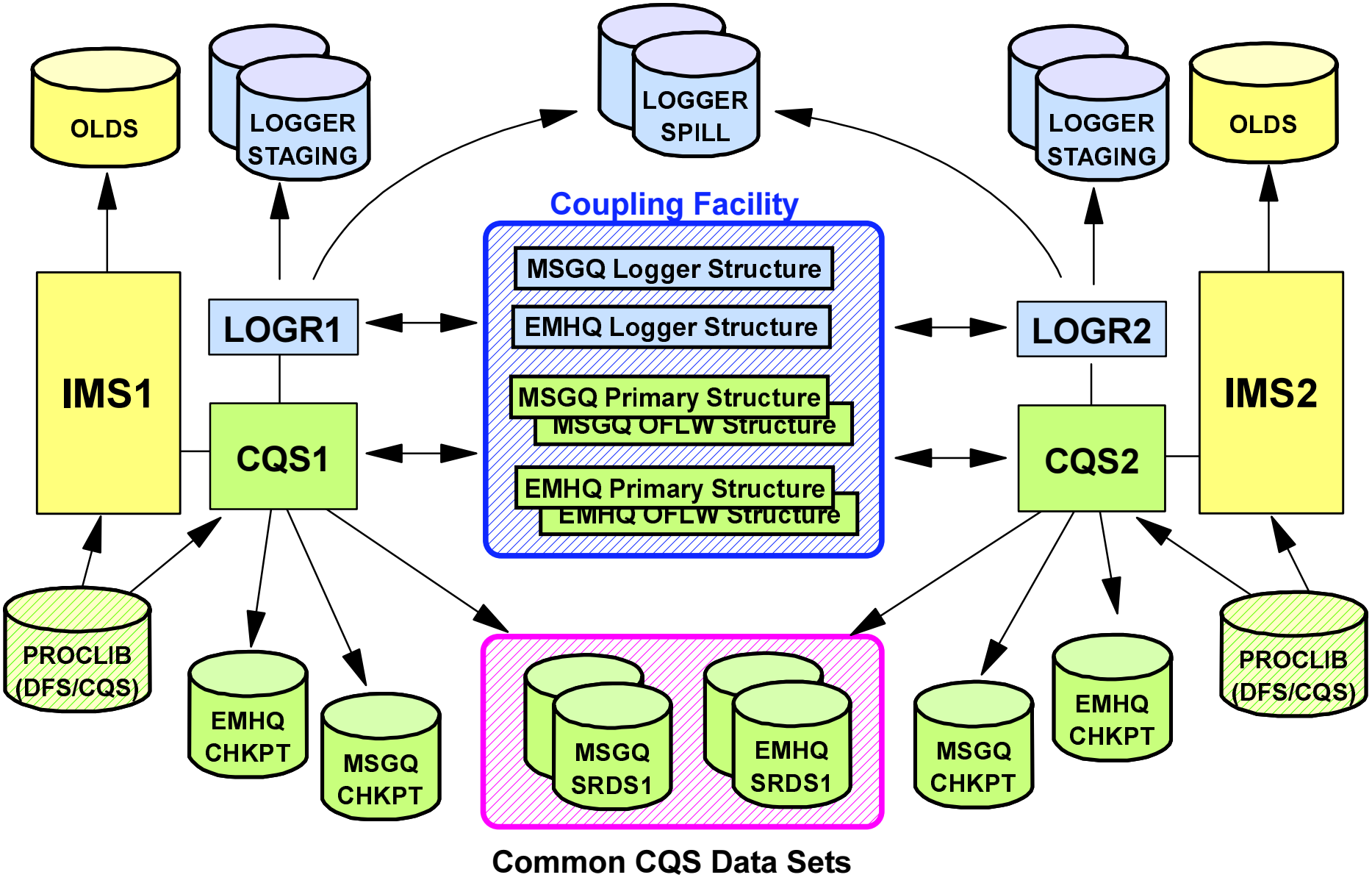
Set by DBFHAGU0 and indicates how transaction is to be scheduled

- If routing code is stopped in front-end IMS, reject the transaction
- **Local Only**
  - If PSB is active in front-end system, queue transaction to BALG
  - Else, reject transaction
- **Local First (Default)**
  - If BALG for PSB in front-end system has five or fewer transactions queued, queue to front-end BALG
  - If PSB is active anywhere in the Sysplex, put transaction on Program Ready Queue for Sysplex-wide processing
  - Else, reject transaction
- **Global Only**
  - If PSB is active anywhere in the Sysplex, put transaction on Program Ready Queue for Sysplex-wide processing
  - Else, reject transaction

Structure Operations

Anaheim, California October 23 - 27, 2000

Shared Queues Components



Structure Operations

Overflow processing

- Dealing with structures in danger of filling up

CQS system checkpoint

- CQS checkpoints its status periodically

Structure checkpoint

- Taking an image copy of a structure

Structure copy

- Making a copy of the structure on the same or another CF

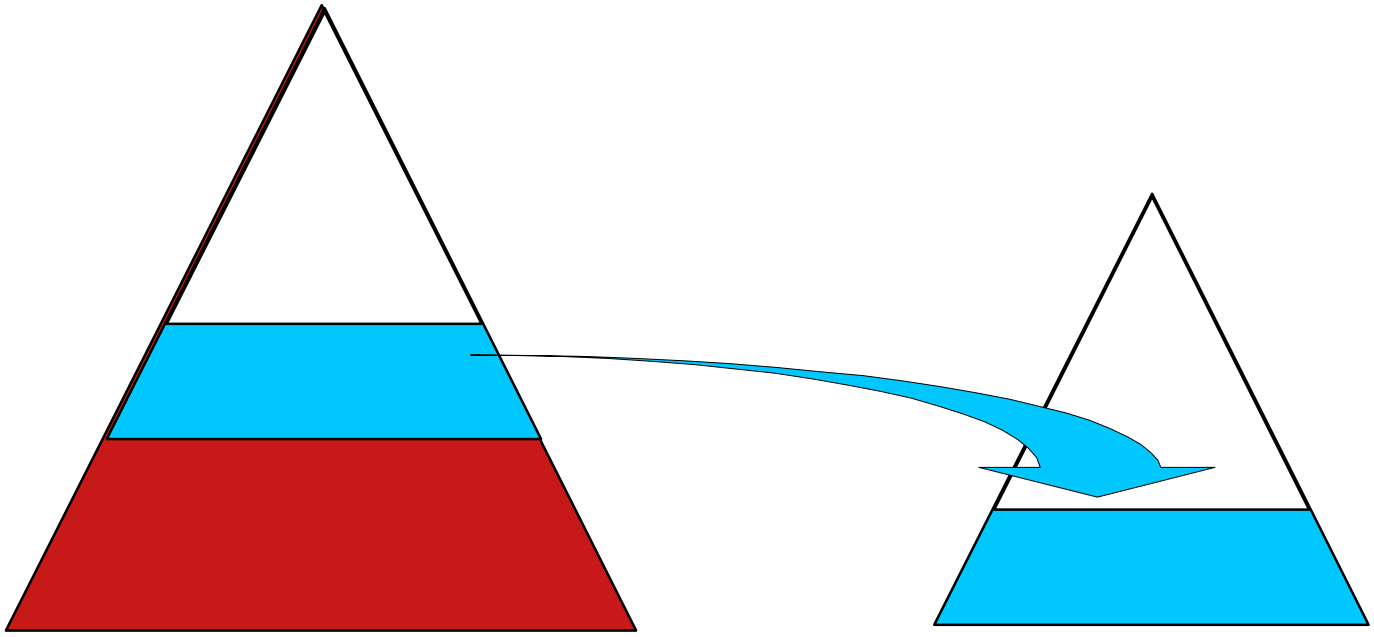
Structure recovery

- Recovery of a damaged structure from the image copy and logs

Structure delete

- Functional equivalent to non-SQ COLDCOMM

Structure Overflow



When Primary Structure reaches a user-defined threshold (percent full), 20% of Primary Structure may be moved to Overflow Structure.

Overflow Processing ...

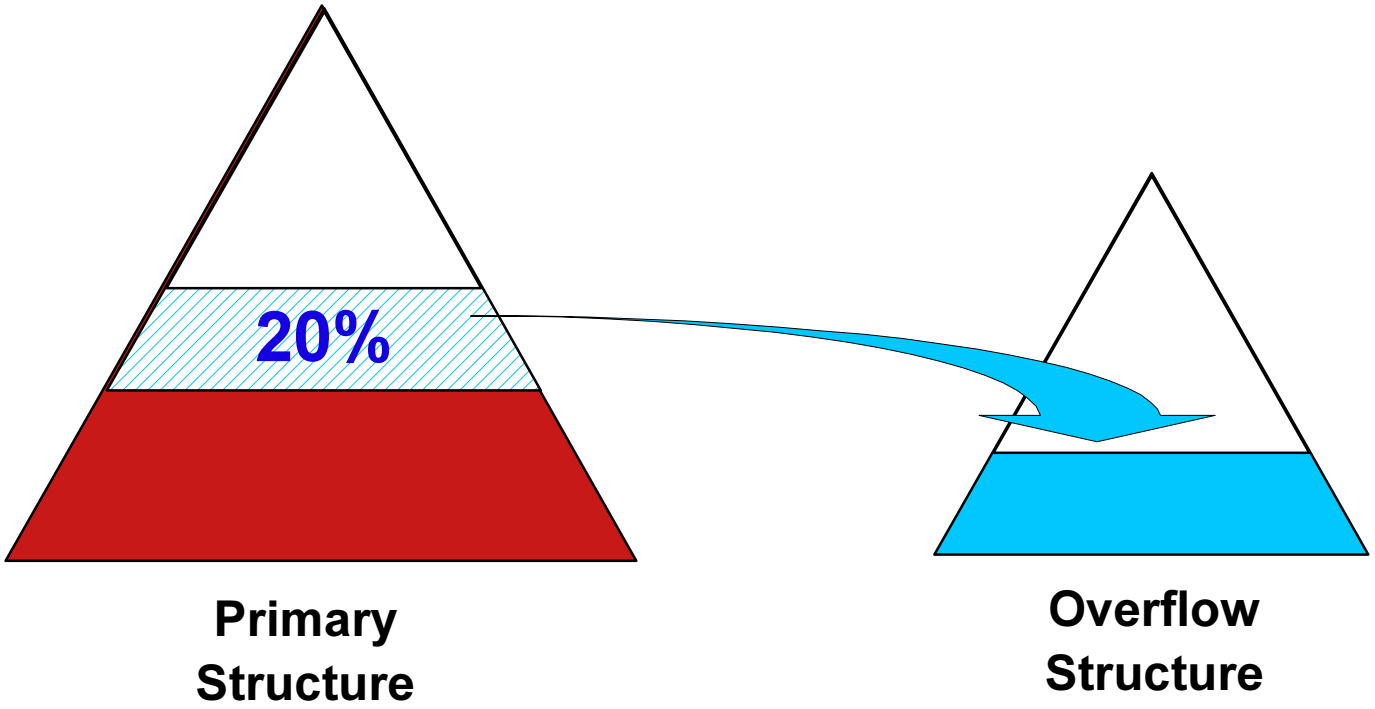
Overflow processing is intended to isolate the offenders

- Invoked when **percentage of data elements in use** reaches a user-defined threshold percentage (Default: 70%)
- For structures defined with INITSIZE parameter in the CFRM Policy
  - If structure size is less than maximum SIZE, **structure size is dynamically increased** to bring utilization down 20%
- **Queues (QNAMEs) using the most data elements** are identified as candidates for overflow processing
  - Structure activity is quiesced while candidates are selected
  - Candidate list represents QNAMEs using 20% of total data elements in primary structure
  - Maximum of 512 candidate QNAMEs
- **User exit can override** CQS decision to put QNAME in overflow status

Overflow Processing ...

Overflow Structure

- Optional
- Contains messages for QNAMEs which have been selected for overflow
- Relieves storage constraint in Primary Structure



Overflow Processing ...

If overflow structure defined ...

- Messages for selected QNAMEs are moved to Overflow Structure
  - These QNAMEs are in Overflow Status
- If threshold is reached again
  - Additional queue names are selected and moved to overflow structure
- Processing continues normally to QNames in overflow status
  - Unless the overflow structure itself becomes full
- **If Overflow Structure becomes full**
  - **PUTs to overflow queues are rejected**
  - **PUTs to primary structure continue until/unless it also fills up**

Overflow Processing ...

If overflow structure not defined ...

- Selected queues remain in primary structure
- PUTs to overflow queues are rejected
- Overflow threshold is not checked again
  - No additional queues are put into overflow mode
  - Primary structure could fill up

Terminating overflow status

- QNAMEs terminate overflow status when the queue is **empty**
- Structure terminates overflow mode when no queues are in overflow status

Structure Full Condition

Either primary or overflow structure may become full

- Out of data elements
- Out of list entries

If either structure is full

- **PUTs to that structure are rejected**
- READs are allowed

Structure full is usually a temporary condition

- DELETEs free up space
- IMS does not stop trying to PUT messages on a full structure

When structures are full, IMS does not abend with a U758.
It continues to attempt to process messages.

Structure Full Condition ...

If PUT is rejected

- Application ISRT is rejected
  - Application gets **QF status code**
- Committed output message not yet on the shared queue structure is **saved in QBuffer (QPOOL)**
  - Retry at next system checkpoint
  - Moved to shared queue when space is available
- Input messages are rejected
  - Non-EMH terminal gets **DFS070** message
  - EMH terminal gets **DFS2194** message
  - MSC link is stopped, **DFS2140** issued by partner
  - System console get **CQS0205E** message (once only)

DFS070	**UNABLE TO ROUTE MESSAGE**
DFS2194	**SHARED EMHQ NOT AVAILABLE**
DFS2140	**DESTINATION name STOPPED, REASON CODE xxx**
CQS0205E	**STRUCTURE xxxxxxxx IS FULL**

If Queues Are Getting Too Full

What can we do about it?

- Several commands available to monitor and correct
  - /DISplay and /DEQueue commands
    - With Shared Queues, /DEQ command can dequeue transactions
- Can make structure larger
  - Change CFRM policy specifying a larger structure size
  - Activate the policy (SETXCF START,POLICY,...)
    - Policy change is pending until structure is rebuilt
  - Rebuild the structure (SETXCF START,REBUILD,...)
    - Structure is rebuilt with new size specification
    - Structure is quiesced during rebuild
- Consider use of Queue Space Notification Exit (DFSQSPC0)
  - Can reject input if destination is in overflow mode

Note: TRANSACTION is a new keyword for the DEQUEUE command, and is only supported in a shared queues environment.

Structure Integrity

Structure integrity is provided by CQS

- All message traffic is logged
  - CQS invokes the **MVS System Logger**
  - All CQSs share the same logstreams
- CQS takes **system checkpoints**
  - Used to restart CQS and resynchronize with IMS
  - CQS checkpoints are written to the MVS System Logger
  - Structure activity is not quiesced
- CQS takes **structure checkpoints**
  - Copy of all recoverable messages on structure pair written to Structure Recovery Data Set (SRDS)
  - Note: Fast path input messages are not recoverable
  - Structure activity is quiesced
- CQS performs **structure recovery** when necessary

Structure Rebuild

MVS supports two types of structure rebuild

- Structure copy
- Structure recovery

Can specify REBUILDPERCENT in CFRM Policy

- Structure is rebuilt when percentage of systems losing access to a structure equals or exceeds this value
- At least one CQS must be active for rebuild

```
STRUCTURE  NAME(MSGQSTR)
           SIZE(100000)
           INITSIZE(50000)
           REBUILDPERCENT(10)
           PREFLIST(CF01 CF02)
```

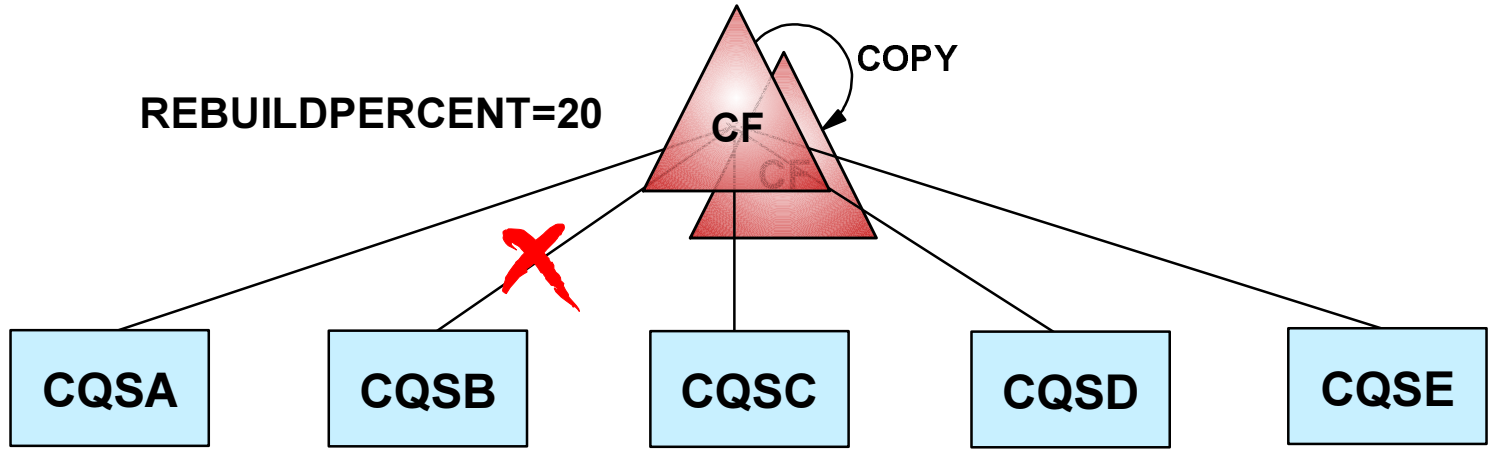
Structure Copy

Copies structure to the same or another CF

- At least one CQS must have access to the structure
- Both recoverable and non-recoverable messages are copied
  - Fast path input messages (for example) are non-recoverable

Automatically invoked when REBUILDPERCENT equaled or exceeded

Manually invoked by command



Structure Recovery

Required if no CQSs can access the structure

- Coupling facility failure
- Structure failure (damage)

Structures are recovered from the most recent Structure Checkpoint (SRDS1 or SRDS2) and merged CQS logstream

- Only full function messages and committed EMH-driven output messages are recovered
- One CQS plays the role of master
  - All CQSs connected to structure participate in rebuild

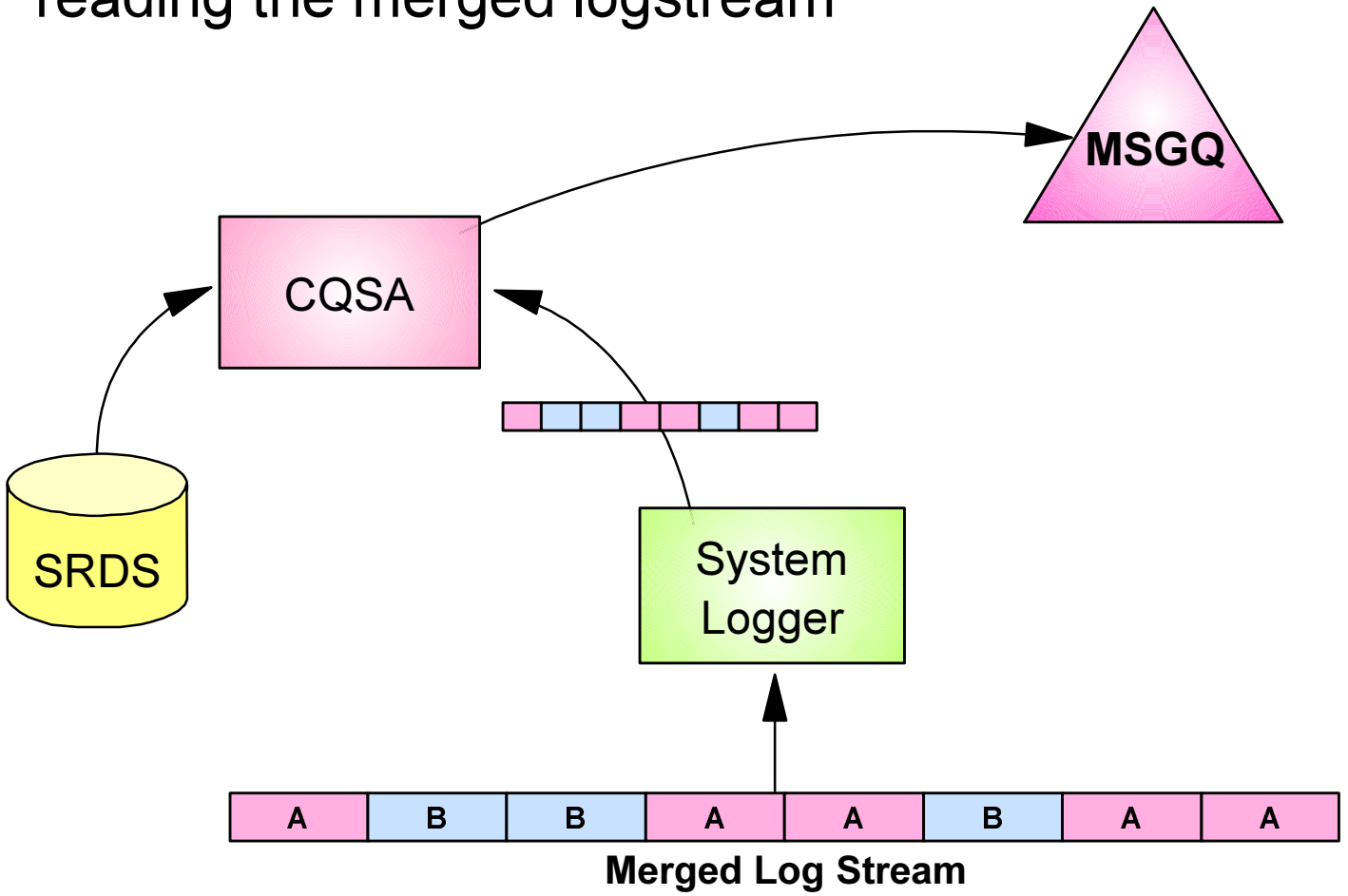
Structure activity is quiesced during structure rebuild

- True for both copy and recovery
- EMH supports option to allow work during structure recovery
  - WAITRBLD=Y can be specified in DFSSQxxx PROCLIB member

Structure Recovery ...

Most recent SRDS is used

- Identifies location in log stream to begin recovery
- **Any CQS** has access to **ALL log records** by reading the merged logstream



Structure Alter

Automatically invoked by CQS if ...

- Overflow threshold is reached and structure is not at maximum size
- For example, if current size is 70000
  - Can increase size up to 100000

```
STRUCTURE    NAME(MSGQSTR)
             SIZE(100000)
             INITSIZE(50000)
             REBUILDPERCENT(10)
             PREFLIST(CF01 CF02)
```

Manually invoked by operator command

```
SETXCF  START,ALTER,SIZE=80000
```

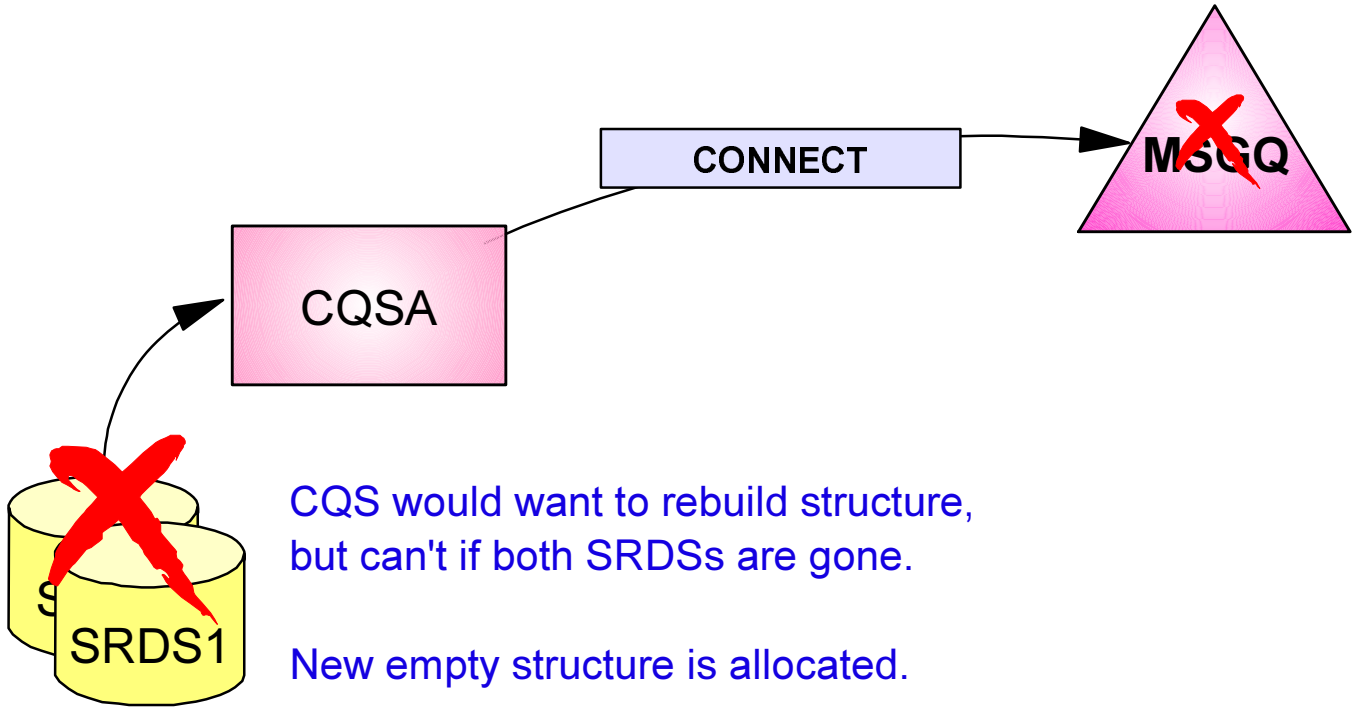
Structure Delete

Why delete a structure?

- Functional equivalent to IMS cold start (COLDCOMM)

How to delete a structure

- Must not allow CQS to recover it when CONNECTing
  - **Scratch both SRDSs**
  - **SETXCF FORCE,STRUCTURE,STRNM=MSGQ**



Shared Queues Summary

Anaheim, California October 23 - 27, 2000

Topics Discussed

Overview of shared queues

- Included a brief comparison to non-shared queues environments
- Discussed concepts of the shared queues function

Interactions between IMS, CQS and the Shared Queues

Shared queues structures

- Characteristics
- Queue types, queue names
- Queuing of messages

Accessing the shared queues

Structure operations

Benefits of Shared Queues

Automatic work load balancing

- A message placed on the Shared Queues can be processed by any IMS with interest in the message

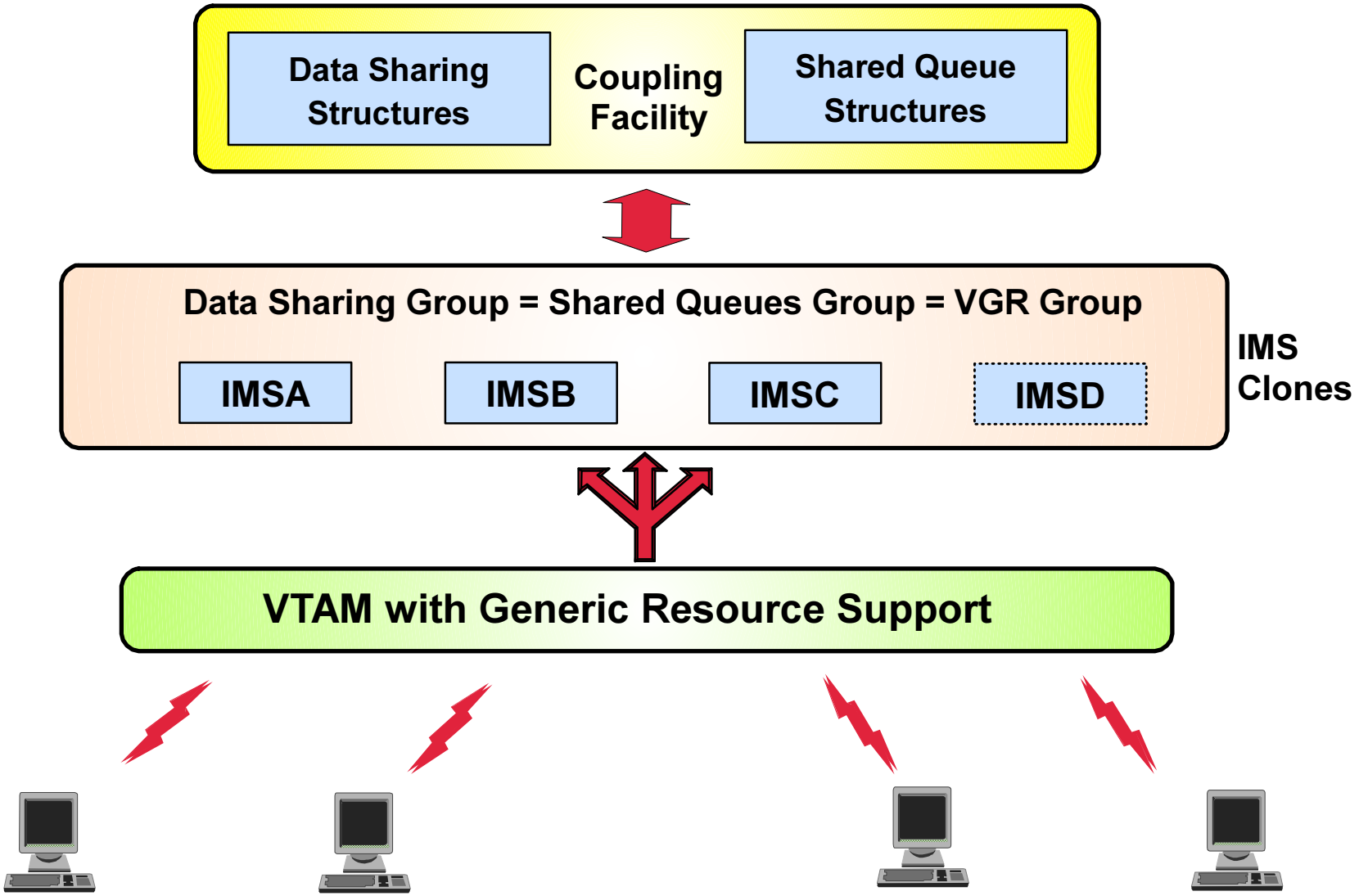
Incremental growth

- New IMS subsystems can be added as workload increases
- New IMSs can be for processing only (no network) during periods of heavy activity

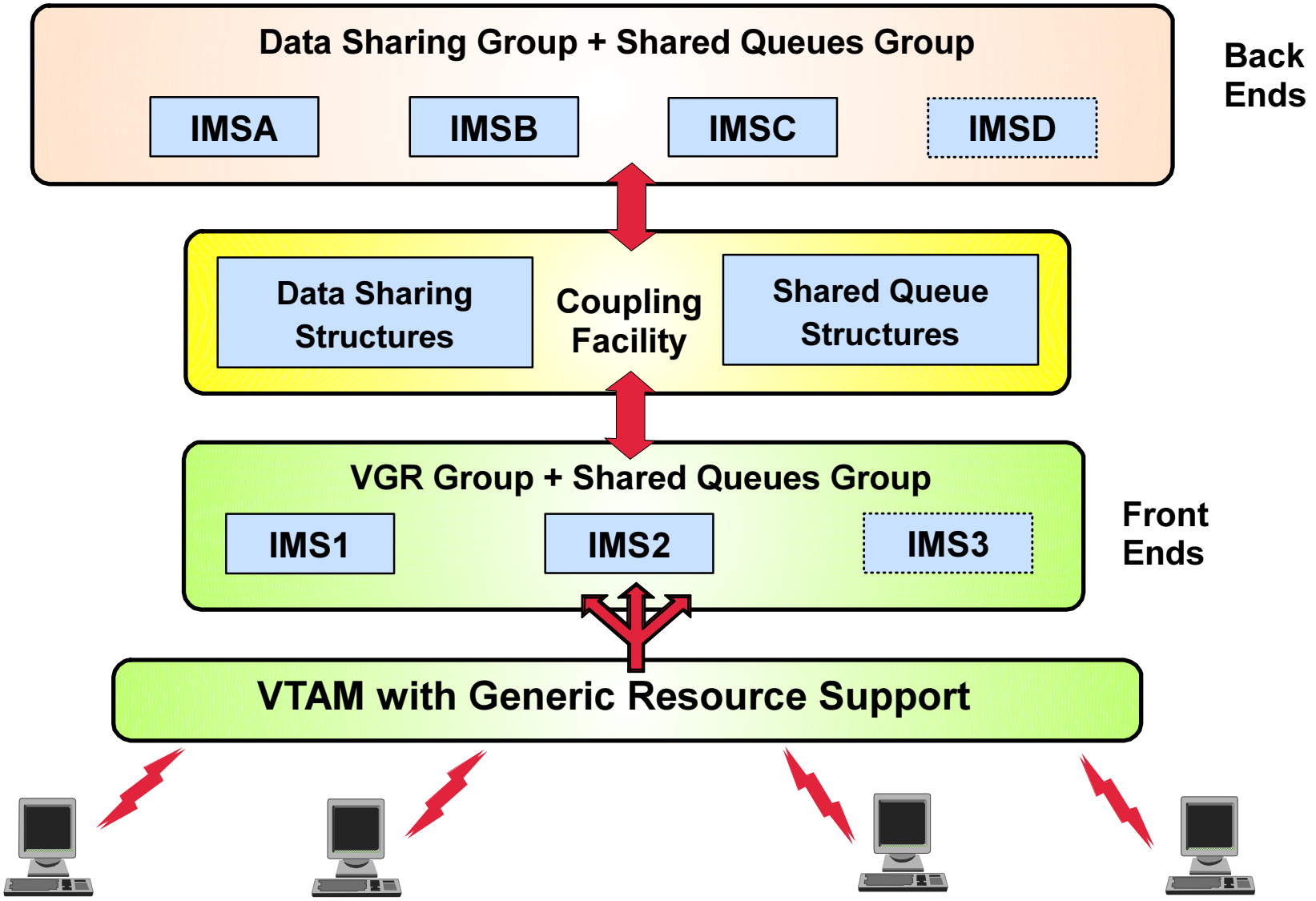
Improved availability

- If an IMS fails, the workload may be assumed by the surviving IMSs
- Shared queues are not lost if one or more IMSs are cold started

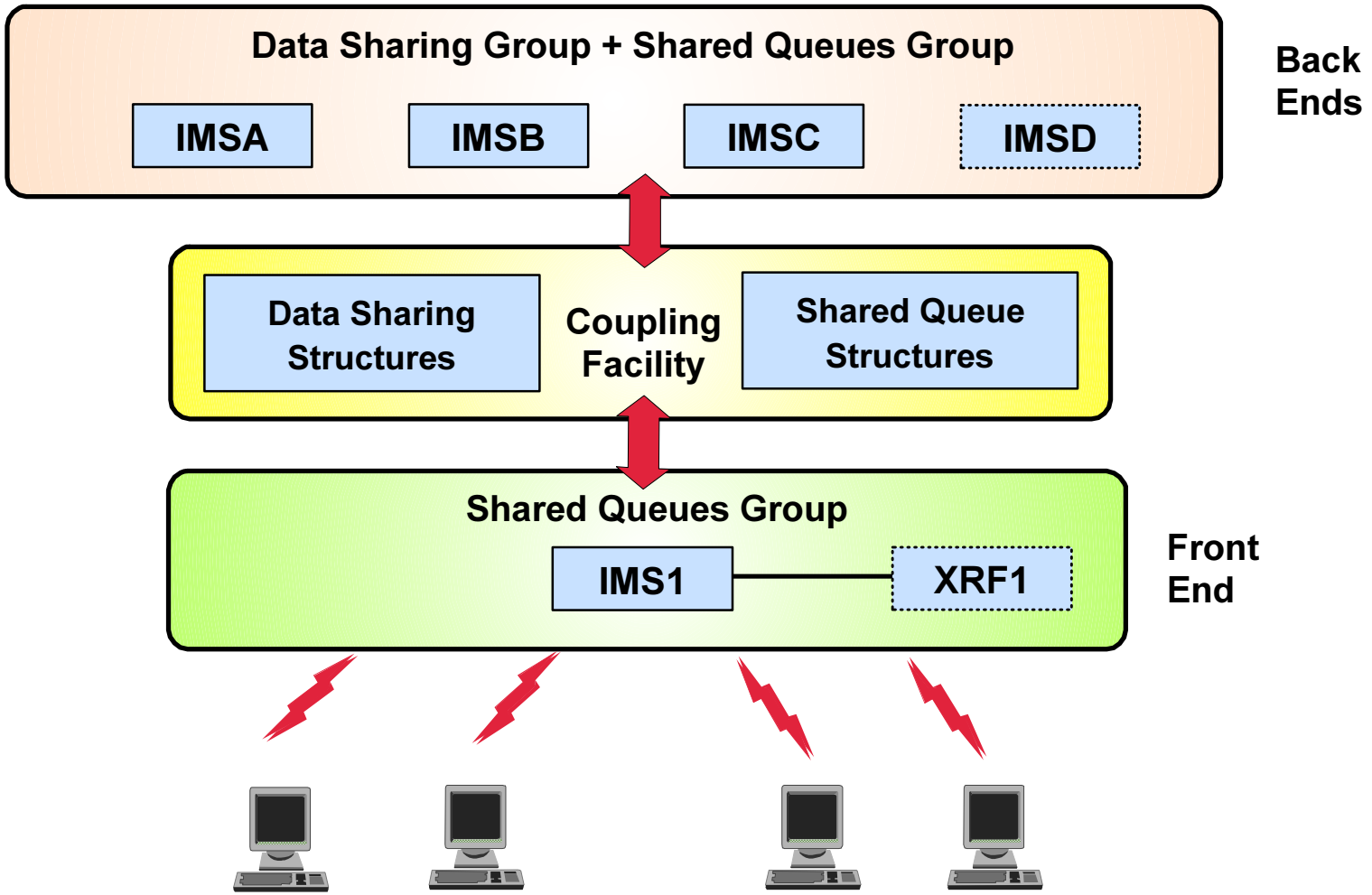
Configuration 1: Cloned IMSs



Configuration 2: Front-End / Back-End



Configuration 3: Front-end w/XRF / Back-end



IMS V7 Enhancements to Shared Queues

Anaheim, California October 23 - 27, 2000

IMS V7 CQS - Multiple Client Support

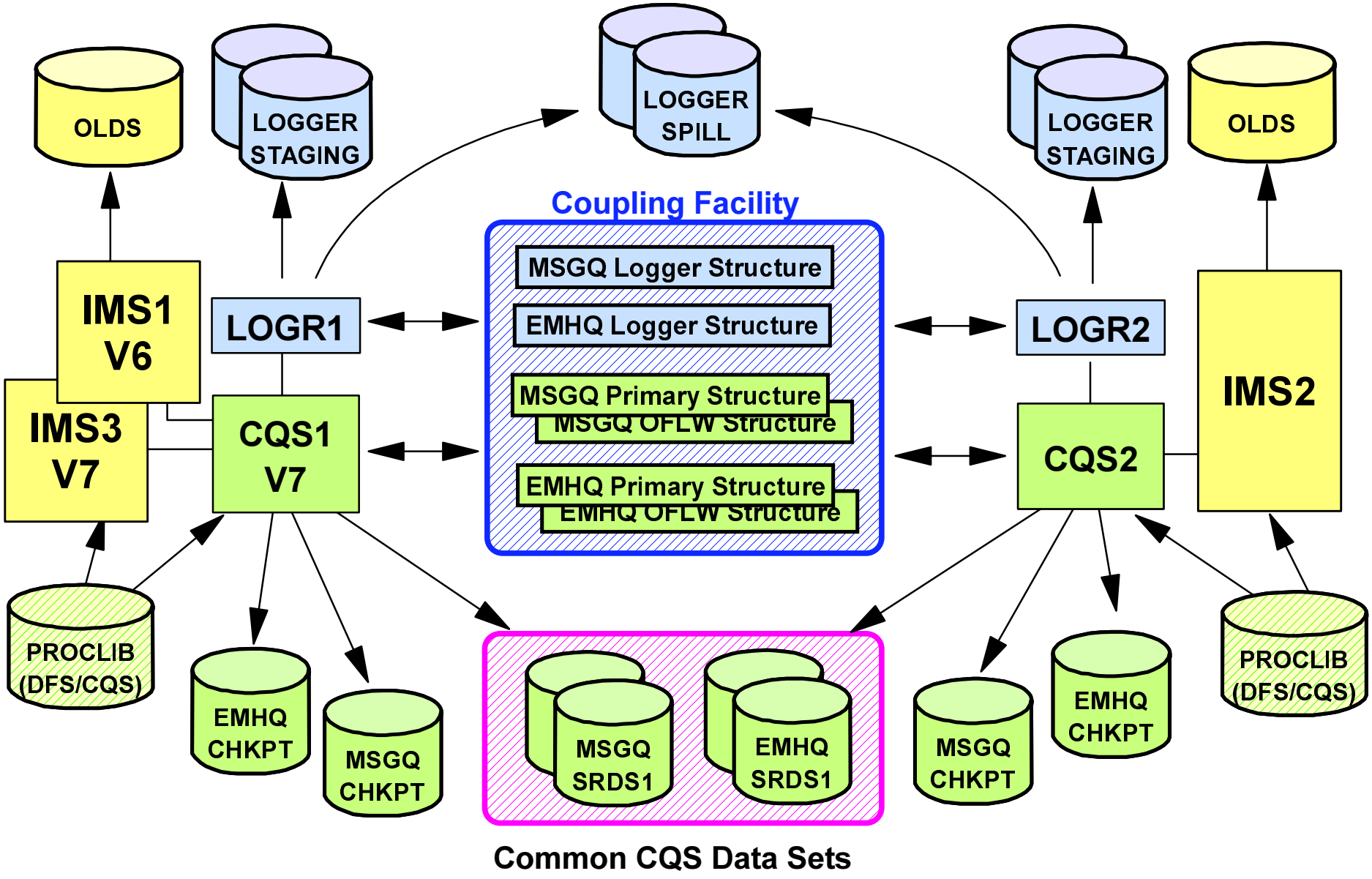
Allows up to 32 clients to use the same CQS address space (all on the same MVS)

- Clients can be a mix of IMS control regions and other clients
- IMS clients can be at different release levels
  - IMS V6 and IMS V7 can attach to the same CQS
  - IMS starts CQS if it is not active
    - Otherwise, IMS registers with the active CQS
    - CQS registration is controlled via RACF or an equivalent product

Migration

- V6 and V7 CQSs can connect to the same CF structures
- Both releases can run on the same CEC

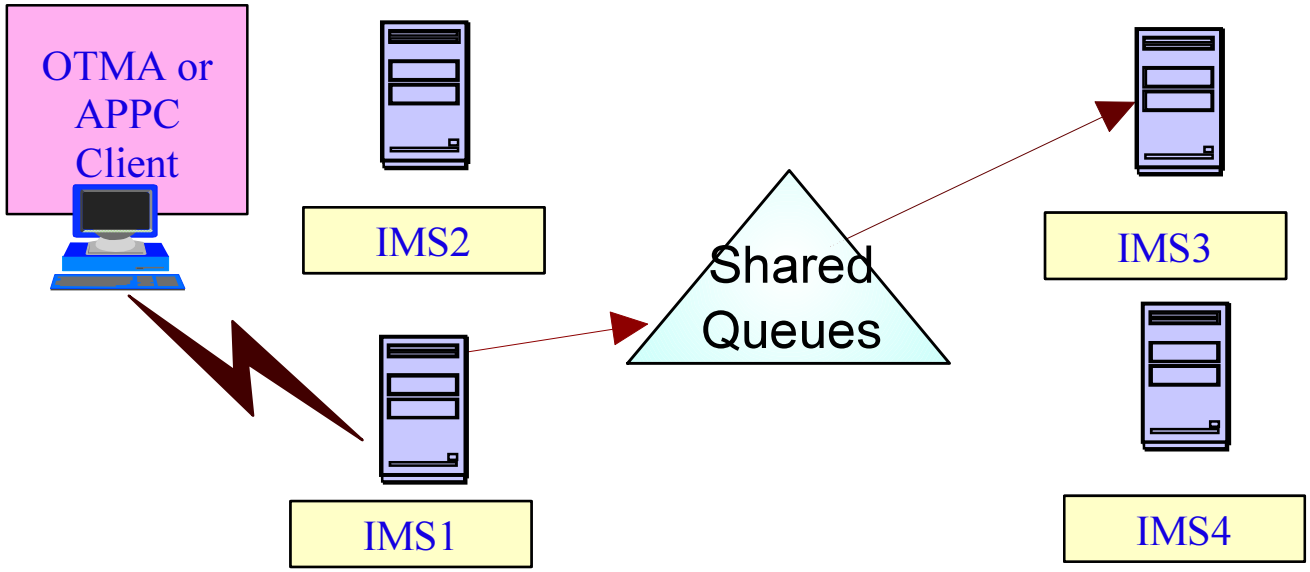
SQ Components - Multiple CQS Clients



APPC/OTMA Enhancement

***Asynchronous* OTMA/APPC input messages are allowed to process on any IMS system in the shared queues group**

- **Assumes** APPC/OTMA are enabled on all back-end systems



***Asynchronous* messages**

- OTMA : Commit - then - send (Commit mode 0)
- APPC : Allocate - send - deallocate

Other V7 Enhancements

RETURN TO INDEX

RACF supported CQS client registration

- CQS rdefined in FACILITY class

Diagnostic enhancements

- New trace entries

See IMS V7 presentation

- ✓ **E68 - IMS V7 TM Enhancements**

IMS Performance Analyzer V2 (V6 and V7)

- Provides Shared Queues information from merged logs