z/TPF Detailed Summary
z/TPF Middleware – IBM MQ

John Muller
z/TPF Development

IBM MQ
Provides

- Enterprise-grade messaging capabilities
- Move information between applications
- Guaranteed one time delivery of messages
- Mature product, 25+ Years

IBM MQ
Messaging Platforms

IBM supported platforms
– Servers
  • Windows, Unix, Linux, z/OS, z/TPF
– Clients
  • Windows, Unix, Linux, **Java**, z/TPF

Other platforms supported by 3rd party such as Willow Technologies
– Mac OS X, BSD, OpenVMS, SCO, SGI, …..

IBM MQ

Message Queue Interface (MQI)

z/TPF does not support all available options and features.
We provide building blocks of primary functions.

MQCONN
MQDISC

MQOPEN
MQCLOSE

MQINQ
MQSET

MQCMIT
MQBACK

MQGET
MQPUT
MQPUT1

IBM MQ
Queue Manager (QMGR)

- Processes application MQI requests
- Contains:
  - Queues
  - Channels
  - Processes

IBM MQ
Queues

A queue is a collection of MQ messages.

A queue is owned and managed by a Queue Manager

A queue contains attributes that control the behavior of the queue and applications
- Maximum queue depth
- Trigger options
- Default persistence of messages
- Message eventing (Queue depth Hi/Lo/Max)
- Put/Get - enabled/disabled

Applications connect to the owning queue manager using MQCONN and then open the queue using MQOPEN, then messages are transferred using MQPUT/MQPUT1 and MQGET.

When finished applications issue MQCLOSE, and then MQDISC

IBM MQ
Queue Types

Local

- On the same computer as the manager.
- Normal
  - Processor Shared for Loosely Coupled (L/C) queues (COMMON=YES on DEFINE QL called TO2 Queues)
  - Processor Unique non-L/C queues (COMMON=NO on DEFINE QL called Turbo Queues)
- XMITQ - the Transmit Queue
  - Internal to the MQ process
  - A processor-unique memory queue

Remote

- Administered by a queue manager other than the physically local one.

Alias

- A name to be used by the application to access a queue other than the defined name.

IBM MQ
Process

A process contains information about an application that can be started using triggers.

A process can be associated with one or many queues.

IBM MQ
Channels

An MQ Channel is a telecommunications connection.

MQ Channels are defined with a pair of channel definitions, one for each end of the channel.

There are two types of MQ channels:

- Client to a remote server. (Message Queue Interface - MQI))
  - CLNTCONN client channels
  - SVRCONN server channels
- Queue manager to queue manager. (Message Channel Agent - MCA)
  - SENDER channels send messages to a remote queue manager
  - RECEIVER cannels receive message from a remote queue manager

MQ channels support SSL for authentication and protection of data
- SVRCONN, SENDER, RECEIVER

IBM MQ
MQ API

MQCONN/MQDISC (Qmgr)
- Associates an application with a Queue Manager that will execute the application's MQI calls, return a connection handle.
- Dissociates an application from a Queue Manager by invalidating the connection handle.

MQOPEN/MQCLOSE (Queue/Qmgr/Process)
- Associates an application with an object by returning an object handle.
- Dissociates an application from an object by invalidating the object handle.

MQGET (Queue)
- Retrieves one message from the top of the queue (FIFO)
- If the MQGMO_WAIT option is specified, the program is reactivated on receipt of a message, or after a TIMEOUT with no message available.

MQPUT/MQPUT1 (Queue)
- places one message at the end of all messages on the queue.
- Issues an MQOPEN, MQPUT, MQCLOSE in one MQI call.

MQINQ/MQSET (Queue/Qmgr/Process)
- Inquires queue or queue manger attributes and settings.
- Sets queue or queue manager attributes or settings.

IBM MQ
Messages

A Message is a collection of bytes, developed by one application that may be delivered asynchronously to another application.
- The message may have any format
  - (binary, character (ASCII/EBCDIC), or a combination

The attributes of a message are described by an MQ Message Descriptor (MQMD) which is added to the message before it is placed on the queue

Messages are placed on a queue with MQPUT or MQPUT1.

Messages are removed from a queue using MQGET

Messages may be persistent or non-persistent.

Messages may be time sensitive. (Expiry)

MQ provides guaranteed one-time delivery of messages

IBM MQ
z/TPF Local Queue Manager (ZMQSC)

The TPF Local Queue Manager allows queues to exist on the TPF system.

With this code, processing of MQI calls may be done in TPF rather than on a physically remote system.

One queue manager per subsystem is allowed.

This queue manager may handle:
- Access to its queues by TPF local applications
- Access to its queues by physically remote applications via an MQI channel (SVRCONN)
- Access to other managers' queues via MCA channels (SENDER/RECEIVER)

One Dead Letter Queue is assigned to the queue manager.

IBM MQ
z/TPF Local Queue Manager (ZMQSC) - continued

The queue manager depends on system heap for memory tables.
- Requires an allocation of 1MB Frames in the 31-bit System heap.
- Requests are made for 4K frames.

In most cases, the queue manager depends on TPF Collections Support (TPFCS) for file space.
- Requires pool records or fixed file records

Turbo MQ (a unique design within TPF) requires:
- Message data is store in SWB's
  - Large messages will require many SWB's
- Fixed file MQ Checkpoint records
  - Enough for two copies.
- MQ uses the recovery log, the commit log fixed file area.

IBM MQ
Only on z/TPF

Free Local Queue Manager/Server
- Selective functionality
- Use MQ Client for full functionality

Processor Shared (L/C) TPFCS queue (COMMON-YES)

Turbo MQ - Memory Queue (COMMON-NO)
- Slow queue sweeper which moves messages from SWB's to TPFCS records

TPF triggers

TPF trigger user exit (CUIR - default with no process defined)

MQ Bridge (CUIR)

Dynamic Routing

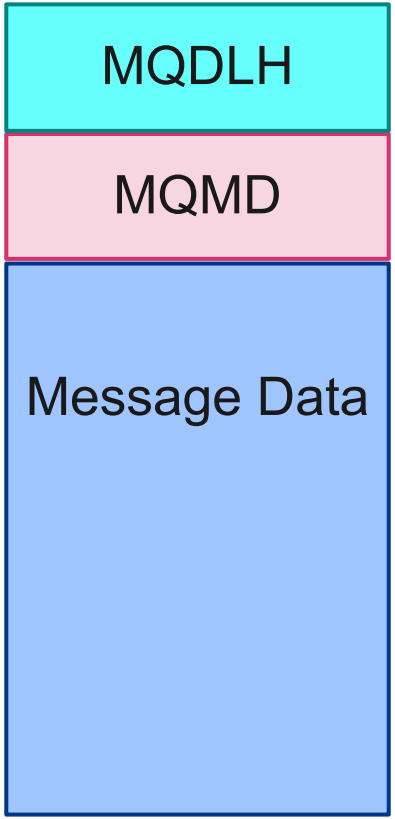
IBM MQ

Queue - Dead Letter Queue

The Dead Letter Queue is a special queue used to intercept messages that cannot be delivered to the requested destination.
- Destination queue is unknown
- Destination queue is full.

It is up to an application to retrieve, analyze, and redeliver the messages on the dead letter queue
- The systems code only does an MQPUT, as it's up to an application or utility to issue the MQGET

The user message is preceded by a Dead Letter Header (MQDLH) when a message is retrieved from the dead letter queue.
- Otherwise, the queue is treated like any other queue.



IBM MQ
Queue - Transmission Queue

The transmission queue is a special, local, processor-unique queue assigned to a message sender channel.

The transmission queue is used as an intermediate local storage facility so that the application may run independent of the operation of the channel.

The transmission queue may "swing", i.e. change loyalties to a different sender channel.

Applications do not issue MQPUT/MQGET from the Transmission queue, instead the messages are placed to the Remote Queue, which ends up on the transmission queue.

The transmission queue operates transparently to the application.

IBM MQ
Queue -Triggers

A trigger causes a program to be activated when a certain activity is detected.

There are two types of trigger in TPF
- TRIGTYPE-FIRST on queue definition
  - MQGET’s issued to empty queue will “set" the trigger
  - A message arrives on an empty queue creates a trigger ECB
- TRIGTYPE-EVERY on queue definition
  - Each message arrival results in a trigger ECB
- Control is passed to the program defined in the process associated with the queue

This is different from MQ Architecture where:
- The trigger message is put onto an initiation queue.
- The trigger monitor gets messages from the initiation queue and takes action.

IBM MQ
Queue - Browsing and Searching

TPF's MQ supports browsing a queue for messages with certain attributes

The queue must be opened with the Browse option
- This establishes a cursor to indicate this ECB's position in the queue for searching.

The queue may then be searched using MQGET with various options
- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

The message may then be removed from the queue using MQGMO_MSG_UNDER_CURSOR

A queue can be searched for a specific message by using MatchOptions in the MQGMO structure and providing the search criteria in the MQMD.
- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID

IBM MQ
Queue – Queue depth monitor

The TPF system provides a queue depth monitor for processor-unique queues.

When the queue depth exceeds specified values, a warning message is sent to the operator console every *xx* seconds.

- MQSC0512W QUEUE qname HIGHWATER MARK nummsg WAS EXCEEDED BY numover
  - qname - Name of the queue
  - nummsg - The number of messages of the high water mark.
  - numover - The number of messages on the queue over the number of the high-water mark
- This could mean that the queue is stalled and may need operator intervention.
- When MAXDEPTH is exceeded applications will receive error Queue is full.
- Maximum queue-depth values and warning time are specified by the QDEPTHHI, MAXDEPTH , and QDT parameters of the ZMQSC DEF/ALT QL or ZMQSC DEF/ALT QMGR commands

IBM MQ
MQ Client Support (ZMQID)

The MQI calls (functions) are executed on a remote MQ server.

The TPF MQI Client channel may be APPC or TCP/IP.

The ECB is suspended while the function is performed on the server.

TPF client supports asynchronous MQGET
- When issuing an MQGET with wait, many ECB's can consume resources.
- Issuing the unique TPF_MQGET MQI and subsequent TPF_MQGET_RESUME allows the ECB to exit. Once a response is received a new ECB is created.

IBM MQ
MQ Client Programming

Whenever an MQ function call is issued, an APPC conversation or TCP connection is started
– With APPC, parallel sessions are strongly advised

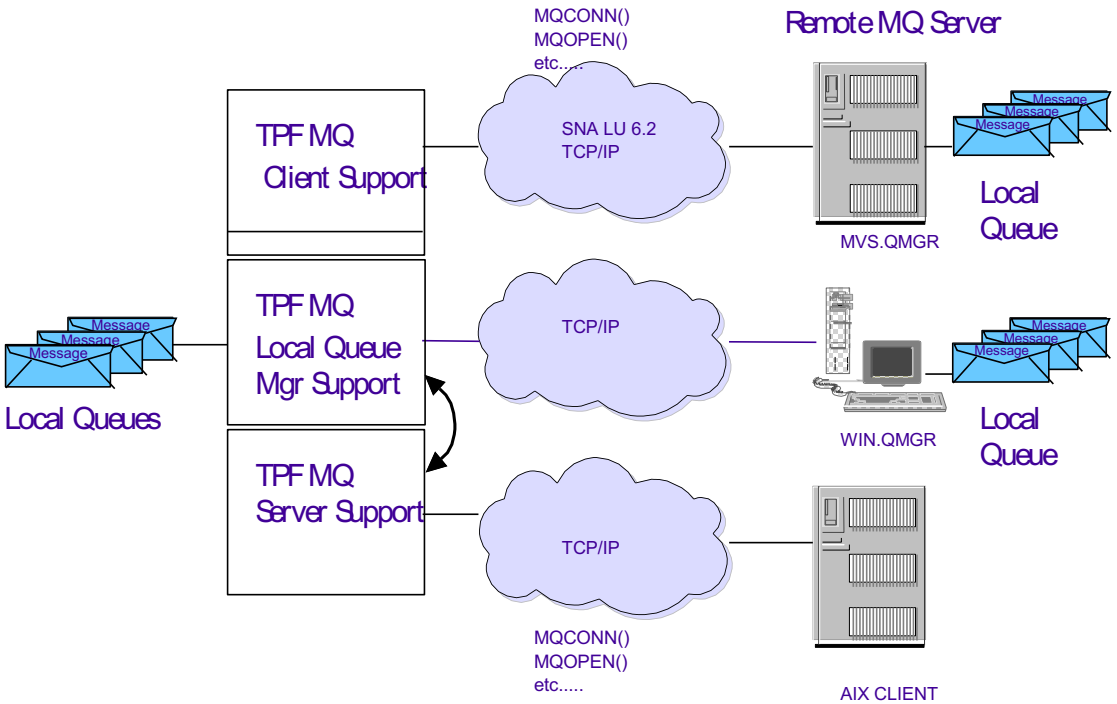
TPF sends data across the MQI channel so that the server will execute the function call.
– Note that TPF does very little MQ processing.

The ECB is suspended and the conversation waits for the return conditions reported by the server.
– This is the main cost in using the TPF Client code: there is a risk of an extended wait by the ECB,
– TPF_MQGET can be used to allow a new ECB to process the message.

With MQ Client, TPF does not have a resident queue of any type.
– Queues are resident on the remote queue manager.

IBM MQ

Client, Server and Local Queue Manager



IBM MQ
Turbo Overview

Processor-unique queues may reside in memory
- Significant performance improvement
  - From about 30/sec to tens of thousands per sec
- Messages in system work blocks (SWBs)
  - SWBs in these queues are included in transaction services commit scopes
- Memory lock for queue updates

Persistence of messages over an IPL
- Checkpointing
- Recovery log

MQ resource manager

IBM MQ
Persistence for Memory Queues

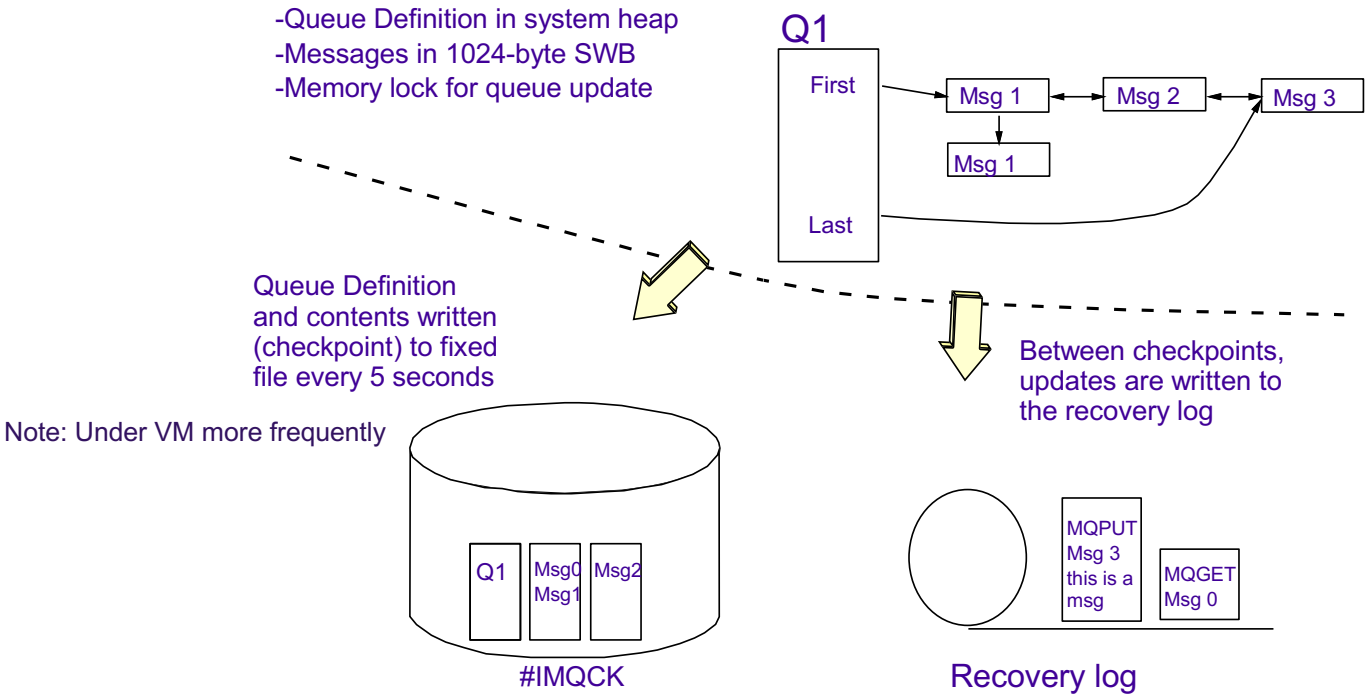
A periodic snapshot of SWB-based memory queue is written to file.
– Checkpointing writes to #IMQCK records

All queue changes between checkpoints are written to the recovery log.
– Full track writes
– Log must be able to accommodate activity between checkpoints
– Only persistent messages are written to the log

During TPF restart, the MQSeries resource manager will apply all log records to the checkpoint to restore the queue to the state it was in before the IPL.

IBM MQ

Persistence for Memory Queues - continued



IBM MQ
Updating memory-resident Queues

MQPUT/MQPUT1 and MQGET

Memory queue is locked for very short period of time

Data is written to the recovery log

MQ local queue manager works with MQ Resource Manager
- MQ APIs work with Transaction Manager APIs
- MQ APIs work in nested commit scopes

IBM MQ
Turbo MQ Performance

The Recovery Log is handled synchronously.
– Any updates to the log suspend the requesting ECB until the log data is written to disk and the interrupt is received.

The Recovery log may handle multiple requests within a single I/O

To accumulate multiple, unrelated pieces of data, the memory buffer associated with the log is not written to the log like normal I/O.
– Data is placed in the buffer and the requesting ECB's address is placed in a table and the ECB gives up control.

When certain criteria are met, the buffer is written to the log.

When the interruption is received, all ECBs in the table are placed on the ready list.

IBM MQ
Nested Commit Scopes

Significant performance improvements can be made by embedding multiple MQGET, MQPUT and MQPUT1 commands in an application commit scope.

MQGET - embedded commit scope example

```
tx_begin - root
        MQGET
        MQPUT
        tx_begin
            MQPUT
            filnc
        tx_rollback
        tx_begin
            tx_suspend
            tx_begin
                    MQGET
                    MQPUT
            tx_commit
            tx_resume
        tx_commit
tx_commit - root
```

IBM MQ
Slow-queue Sweeper

The TPF MQSeries memory-queue sweeper prevents a large memory queue from causing input list shutdown or Resource Depletion Catastrophic SERRC.

Frees SWBs by moving messages that reside in memory to file in TPFCS collections
- The speed at which a queue is serviced determines whether it will be swept
  - If a queue is serviced but more slowly than messages arrive, the queue can build and still run the system out of memory

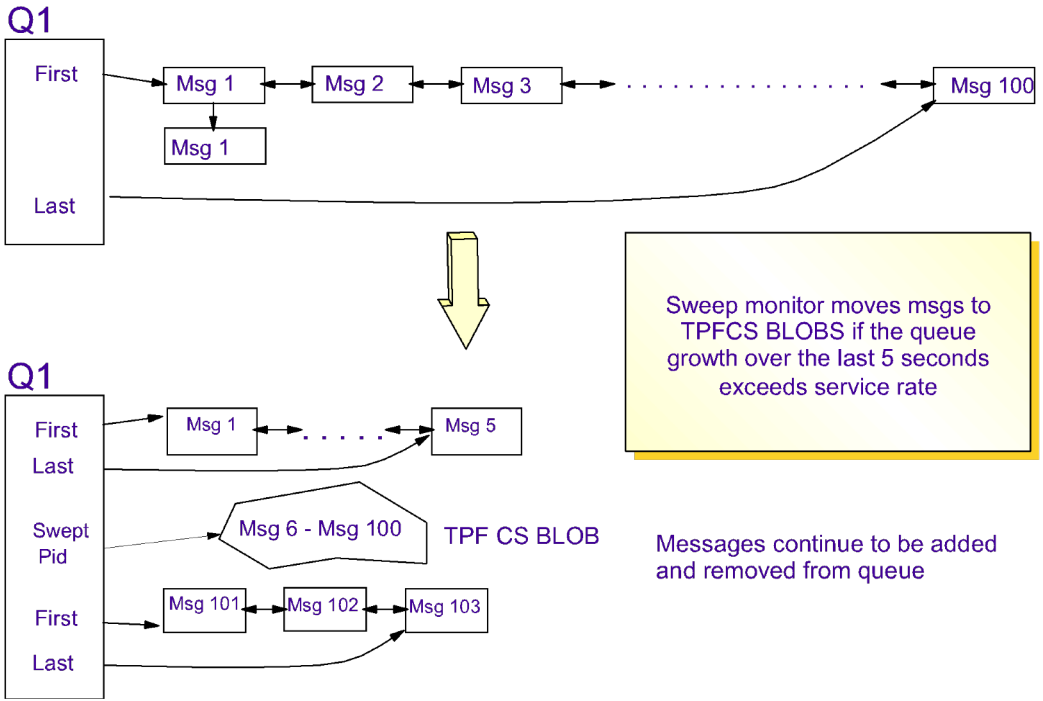
A monitor runs every second
- Checks queue growth against service rate over last 10 seconds
- Moves messages from memory to TPFCS collection and releases SWBs
- Attempts to leave 10 seconds worth of data in memory

Use of the monitor is optional but is strongly recommended

IBM MQ
Slow-queue Sweeper

Queue definition in system heap

Messages in SWB



IBM MQ
Business Events

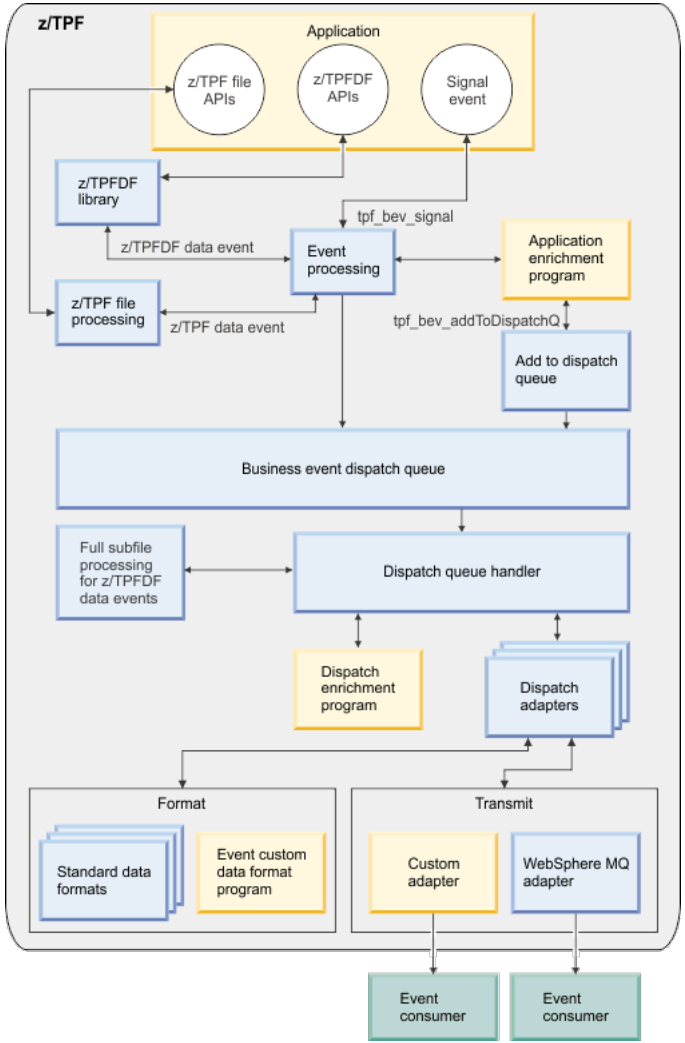
Two types of events:

- Signal Events, (tpf_bev_signal())
- Data Events, modification of data

Under the covers business events uses MQ

- Process - IBEV.DISPATCH.PROCESS, application BEVQ
- Local Queue - IBEV.UNORDERED.DISPATCH.QUEUE, (TRIGGER-FIRST)
- Error Queue – Defined by user

IBM MQ
Business Events



IBM MQ
Dynamic Routing

MQ Remote Queue definitions can be replaced with an MQ Dynamic Routing configuration file.

Remote Queue Definition

```
        Queue Name:  RQ1
      Remote Queue:  Q1
Remote Queue Manager:  TPFQM1
  Transmission Queue:  XQ1
```

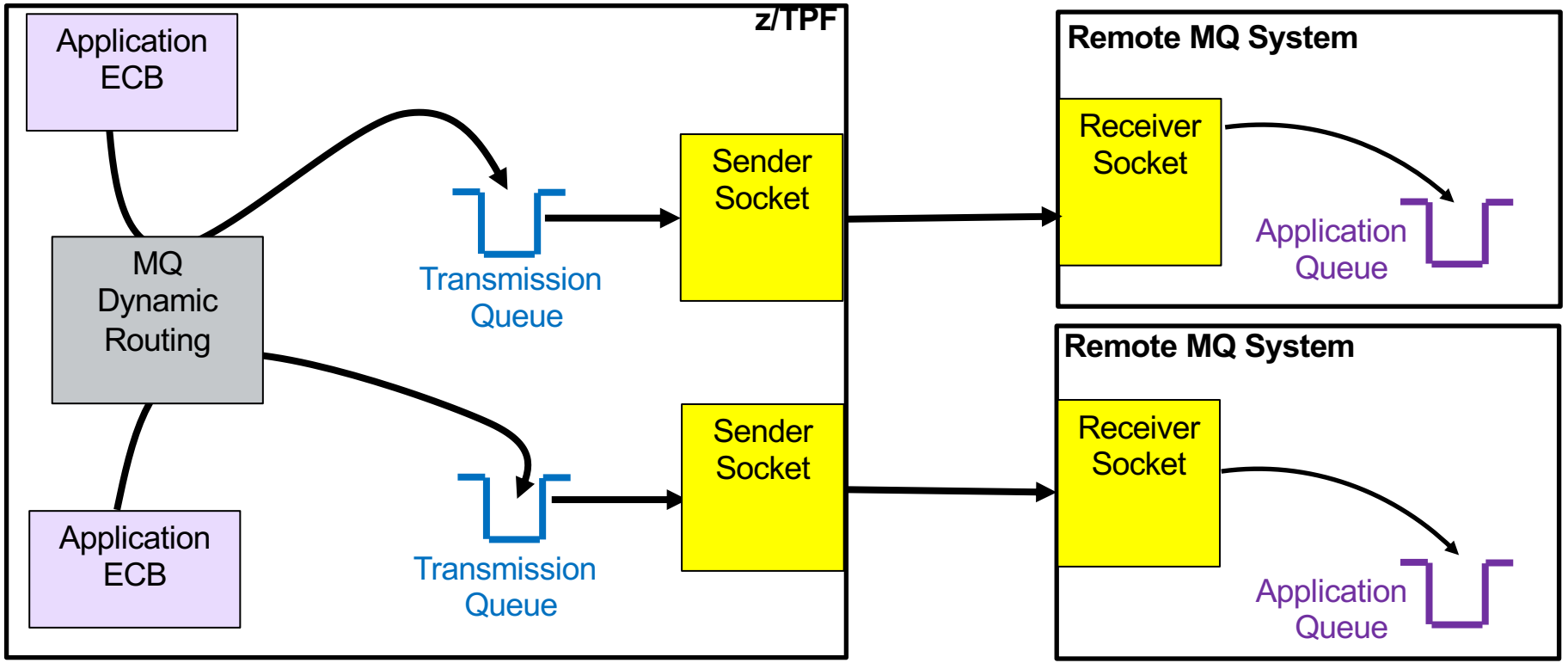
MQ Dynamic Routing Configuration File

```
#Name,  Remote Q,  Remote QMGR,  Transmission Q

RQ1,    Q1,        TPFQM1,       XQ1
RQ1,    Q1,        TPFQM1,       XQ2
RQ1,    Q1,        TPFQM2,       XQ3
RQ1,    Q1,        TPFQM2,       XQ4
```

- When the MQ Dynamic Routing Configuration File is loaded to the z/TPF file system
  - z/TPF automatically detects this and builds the in-core routing table

IBM MQ
Dynamic Routing



IBM MQ
MQI Support Summary

TPF may act as an MQI client
– MQI call executed remotely
– May use APPC or TCP for communications connection.

TPF may act as an MQI server
– A remotely requested MQI call is executed by TPF

TPF may act as an MQ Manager
– MQI calls requested on TPF are executed on TPF

Thank You!

Questions or Comments?

Trademarks

Notes

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.