The following list of symptoms led us to an underlying problem of application requests flooding a coupling facility.

1) MQ Statistics Symptom - The number of message managers gets was much lower than the number of data manager gets.
    a. As we have noted, the message manager counts the number of MQGET requests that come into the queue manager. The data manager reports the number of gets that are used to fulfill the request. When the Data Manager Gets exceeds the Message Manager Gets, that is an early and high-level indicator of message selector use or of queue not being indexed properly. When the MM GET count is higher than the DM GET count that is an indication that GETs are being done against empty queues, so the request is not passed to the DM.
        i. Note – for private queues, a count of skipped messages is kept in the WQ records. There is no equivalent for shared queues.
    b. The Coupling Facility statistics records, QEST, showed a dramatic rise in the number of Coupling Facility redrive requests over the problem intervals.
        i. True for both single entry and multiple entry requests, a relatively small number of redrives is normal but when they exceed 20% during an interval, we try to look into it.
2) Coupling Facility Symptoms
    a. High CF CPU use – high CPU use can be from any number of things including workload spikes. However, it has been observed in several instances of scrolling thru queues at several customers. It requires examination of the Coupling Facility Activity Report, or its equivalent, based on the RMF data.
    b. Sub-channel waits reported – Again, not a specific symptom as there can be a few reasons for this to happen. One is an increase or spike in the number of requests to a specific CF structure, causing contention on the subchannels that provide communication between the LPAR and CF. Other reasons we have seen this happen is due to CF Links being taken offline, typically accidentally, and changing hardware configurations. Sub-channel waits are also reported as part of the Coupling Facility Activity Report
3) MQ Task Accounting Symptoms:
    a. The percentage of valid gets was extremely low for at least one of the queues using the problem structure. In this case the percentage of valid gets for all the queues on the troubled structure was less than 1% in total and the GET count was very high for at least one queue on that structure. That is cause for concern in most environments. Please see 'comparing valid MQGETs' below for additional information.
4) Many MQGETs were for specific messages.
    a. In the WQ records contains counts of each type of MQGET; destructive get for any message (FIFO), destructive get for a specific message, browse for any message (FIFO), and browse for a specific message.
    b. There was no index defined for the shared queue that exhibited the most specific message gets on the problem structure.

c. As described above, these long running TXs do not have the 'select count' indicator set for any interval that does not contain an open.

5) Comparing valid MQGETs with those that do not return data in the WQ records. As stated above, the number of valid gets was extremely low during the problem period. However, this symptom can be caused by application style choices and may not be easy to remedy. Some issues include:
    a. Polling applications – often these are applications that do not use triggering or gets with a wait interval, they repeat the MQGETs until a message is returned. This particular pattern can be made worse by client applications that are poorly behaved (connect, do one thing, disconnect, repeat). It is also the typical implementation pattern from some application enablement tools, so programmers may not even be aware that this is what is happening.
    b. Gets with waits – MQ will signal all instances of an application in a wait state when a new message arrives on a queue. This causes the MQGET request to be resent (as if new). This can lead a high number of 'empty Gets' if the messages are being put at a comparatively slow rate when compared to how quickly messages can be retrieved.
    c. Too many getting application instances – tuning the number of application instances to peak periods can lead to a lot of empty gets during the slower times.
    d. Related to c is the fact that we, the WSC, have seen customers adding instances of the getting applications to address response slowdowns, making the problem much worse. When the slowdown is caused by too few instances of a processing application, adding more helps. When it is caused by internal contention over a particular resource, more instances make the problem worse.
    e. The last symptom we found was when we examined the calls to the CF to fulfill the requests. I had never examined the data at this low a level before, having been told often enough that 'you should never need to.'
    The CF calls made depend on the type of MQGET being done. A simple FIFO MQGET is likely to only use a GETMOVE request to the CF – returning the next message available. An MQGET using a message property match is more complex. What we found was that the requests to get a message by a selector match issued one-to-many READLIST requests for each GETMOVE (the real get of the selected message). When things were running well, the ratio of READLISTs to GETMOVEs was no more than 3 to 1. Most often it was 1 to 1. During the problem period, the number of READLIST requests became exponentially higher than the GETMOVEs. The READLIST requests return a buffer of messages to the queue manager and will set a cursor on the queue to indicate the starting point for the next READLIST in case none of the messages in this group match the selection criteria. The number of messages that can be returned by the READLIST depends on the size of the buffer, the size of the messages, and any 'congestion' that may be taking place within the structure itself. With an increasing depth on the queue, the number of READLISTs increased because the GETs were requesting newer messages. The older messages had an expiration set and that was being honored, the problem would have been much worse if that were not the case. However, the message expiration was set much higher that the MQGET wait expiration, so there were messages on the queue that were never going to be matched and were being read

multiple times.
At the same time, there were applications continuing to put messages to the ‘problem’ queue, and other queues with active puts and gets on the same structure. None of the other queues had an extremely high volume at the time, but there was other activity on the CF structure.

This is a good example of a problem waiting to happen – none of the processing was new or changed by the application or the queue manager. It reached a critical mass due to increased volume of certain types of requests, contention on the queue, and contention on the structure hosting that queue. In my next article I will publish the query that assisted us in finding the CF calls that were used.