UNIVERSITÄT LEIPZIG

Enterprise Computing
Einführung in das Betriebssystem z/OS

Prof. Dr. Martin Bogdan
Prof. Dr.-Ing. Wilhelm G. Spruth

WS2012/2013

WebSphere MQ Teil 3

Trigger

MQ Client/Server Communication

In einer WebSphere MQ Client/Server-Konfiguration arbeiten ein Rechner und sein Queue Manager als Client, und ein anderer Rechner und sein Queue-Manager arbeitet als Server.

Streng genommen stellt WebSphere MQ eine Peer-to-Peer-Kommunikation dar. Es ist jedoch nützlich, den sendenden Queue-Manager als Client und dem empfangenden Queue-Manager als Server zu betrachten, weil WebSphere MQ häufig in einem Request / Response-Modus betrieben wird. Das sendende System erwartet (asnchronously) eine Reaktion von dem empfangenden System beim Senden einer Nachricht (Request-Message), und das empfangende System reagiert durch das Senden einer Response Message.

In diesem Text benutzen wir den Begriff MQ-Client für einen Queue-Manager, der eine Nachricht an eine Target Queue sendet, und den Begriff MQ-Server für einen Queue-Manager, der die Target Queue unterhält, und der potentiell mit einer Antwort-Nachricht an den MQ Client reagiert.

Bitte denken Sie daran, WebSphere MQ ist eine unidirektionale Kommunikation. Um in einem Request / Response-Modus zu arbeiten, muss der MQ Server eine zweite unidirektionale Verbindung vom MQ-Server an den MQ-Client verwalten.

Fire and forget

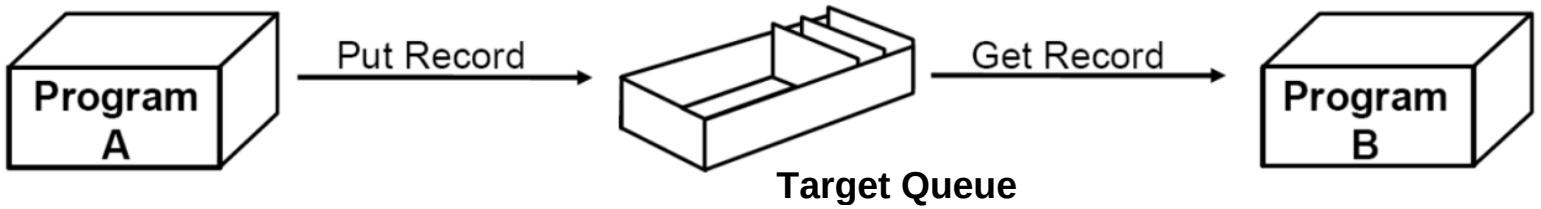


Abb. 3.1

WebSphere MQ ist ein unidirektionales Middleware-Produkt. In seiner einfachsten "Fire and Forget"-Form, überträgt das sendende Programm A eine Nachricht an einen empfangenden Programm B und vergisst die ganze Sache. Es liegt an der empfangenden Programm B, die Nachricht irgend wann von der Target Queue zu extrahieren.

Die Nachricht befindet sich entweder auf dem sendenden Rechner oder dem Target Rechner, je nachdem, ob die Nachricht bereits übermittelt wurde oder nicht. Programm A kümmert das alles nicht.

Wenn das empfangende Programm B Programm nicht verfügbar ist, wird die Nachricht in einer Queue bleiben und später verarbeitet. Programm B kann den ganzen Tag lang warten, ehe es mit einem MQGET Befehl die Nachricht ausliest.

Eine Alternative ist, Programm B automatisch zu starten, wenn eine Nachricht eintrifft (oder nachdem eine bestimmten Anzahl von Nachrichten eingetroffen sind).

Request/Response

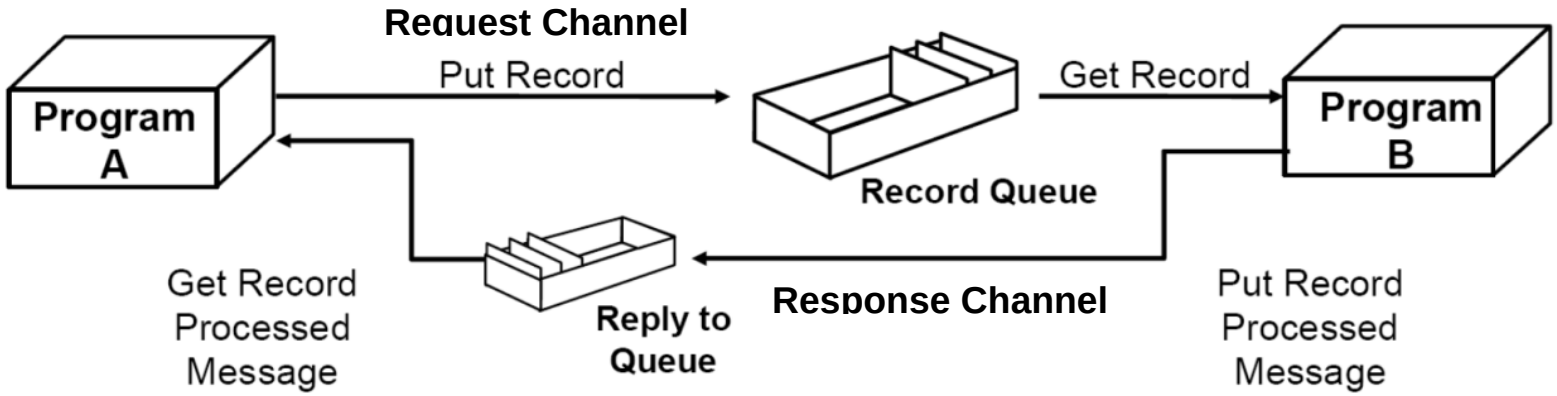


Abb. 3.2

Obwohl WebSphere MQ asynchron arbeitet, kann ein "Request / Response" Modus einen synchronen Betrieb simulieren. Es wird recht häufig auf diese Weise verwendet. Dazu enthält der Message Descriptor der Request Nachricht eine "Reply to ...."-Anzeige.

Da WebSphere MQ asynchron und unidirektional arbeitet, werden zwei MQ-Kanäle mit zwei Message Channel Agent (MCA) Paaren für einen Request / Response Betrieb benötigt.

MQI Channel

Bei den WebSphere MQ Netzwerk-Kommunikationsverbindungen existieren zwei verschiedene Arten von Channels:

Message channel

Ein Message Channel (in der Regel nur als Channel bezeichnet) ist das, was wir bisher diskutiert haben. Er verbindet zwei Queue Manager durch Message Channel Agents (MCAs) auf jeder Seite. Ein Message Channel ist unidirektional, besteht aus zwei Message Channel Agents (ein Sender und ein Empfänger) und einem Kommunikationsprotokoll. Ein MCA überträgt Nachrichten von einer Transmission Queue zu einer Kommunikationsverbindung, und von einer Kommunikationsverbindung zu einem Target Queue. Für eine bidirektionale Kommunikation ist es notwendig, ein Paar von Kanälen, bestehend aus einem Sender und einem Empfänger-Channel zu definieren.

Der Queue-Manager überträgt Nachrichten an andere Queue-Manager über Channels. Hierzu werden vorhandenen Netzwerk-Einrichtungen, in der Regel TCP / IP, benutzt.

Channels sind unidirektional. Die meisten der Zeit verstehen wir unter einem Channel einen Message Channel.

MQI channel

Ein spezieller Fall eines Channels ist der Message Queue Interface (MQI) Channel. Es verbindet einen WebSphere MQ-Client (auch als Slim-Client bezeichnet) mit einem Queue-Manager. WebSphere MQ Slim Clients verfügen nicht über einen eigenen Queue-Manager.

Ein MQI Channel ist bidirektional.

Slim and fat Clients

WebSphere MQ unterscheidet zwischen verschiedenen Arten von Clients:

- **Slim client (oder WebSphere MQ client)**
- **Fat client**

Der Unterschied zwischen einem Slim Client und einem Fat Client besteht in der Art, wie Nachrichten gesendet werden. Die Transmission Queue befindet sich entweder in dem Endbenutzer-Arbeitsplatz (Fat Client) oder auf einem „Slim Client-Server“ (SCS). Ein Slim Client muss an einen WebSphere MQ Slim Client Server angeschlossen sein, der die Queue-Manager Services zur Verfügung stellt. An einen SCS sind in der Regel viele Slim Clients angeschlossen.

Fat Clients haben eine lokalen Queue-Manager, Slim Clients nicht.

Ein Slim WebSphere MQ-Client, der sich nicht mit einem SCS verbinden kann, kann nicht arbeiten, weil sich der Queue-Manager und die Queues für den Slim Client auf dem SCS befinden. WebSphere MQ-Clients sind häufiger Slim Clients und seltener Fat Clients.

Hinweis: Die "WebSphere MQ-Client für Java" ist ein Slim Client.

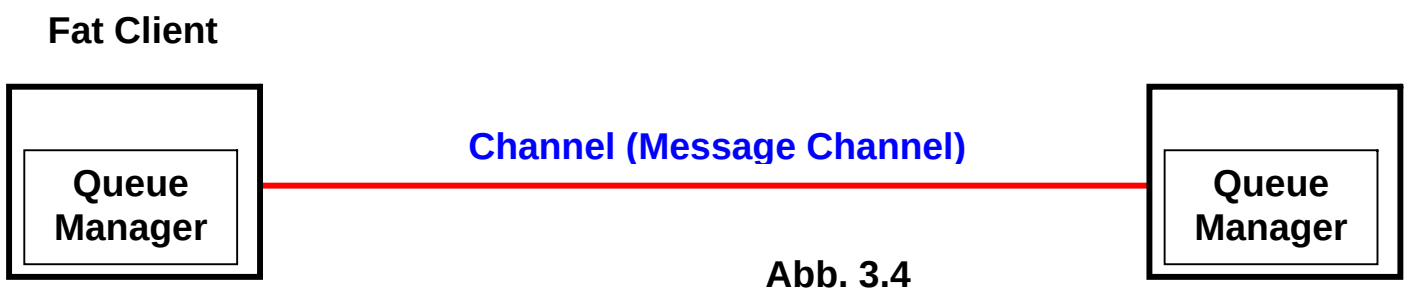


Abb. 3.4

Es existieren zwei Arten von WebSphere MQ Clients: Slim-Clients und Fat Clients. Ein Fat Client enthält einen eigenen Queue Manager. Für eine Gruppe von Slim Clients stellt ein WebSphere MQ Slim Client Server (SCS) Queue Manager Services zur Verfügung.

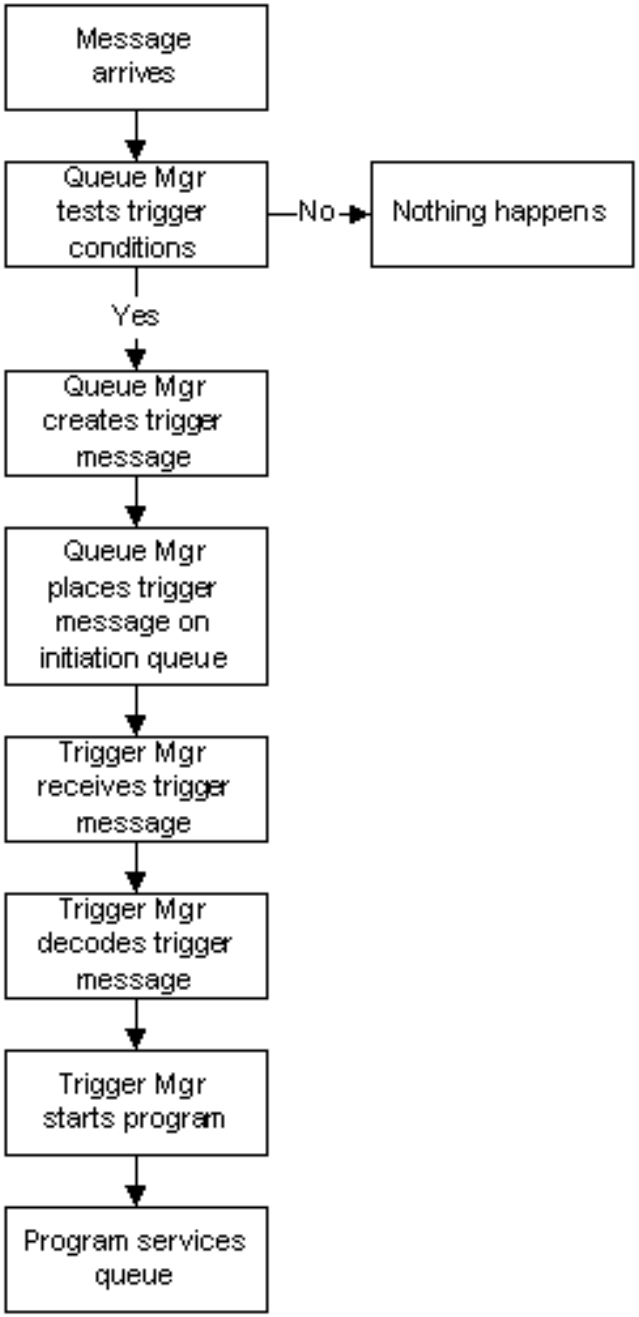
Trigger Event

Die Target Anwendung kann die Nachricht aus der Target Queue zu einem beliebigen Zeitpunkt extrahieren. Dies kann zu einer langen Wartezeit führen, ehe eine Antwort gesendet wird. Um pseudosynchron zu arbeiten, muss die Target-Anwendung "getriggert" werden. Die Target Anwendung wird informiert, dass eine Nachricht in der Target Queue angekommen ist, die ihre Aufmerksamkeit erfordert. Dazu wird ein „Triggering Mechanismus" benötigt.

Ein "Trigger-Event" ist ein Ereignis, das den entfernten Queue-Manager motiviert, eine "Trigger-Message" zu generieren. Ein Trigger-Event wird in der Regel in dem Deskriptor einer Nachricht angegeben.

Wenn Triggerung für eine Target Queue aktiviert ist und ein Trigger-Event auftritt, sendet der Queue Manager eine "Trigger Message" an eine lokale Queue, die als Initiation Queue bezeichnet wird. Das Vorhandensein der Trigger-Nachricht in der Initiation Queue zeigt einen Trigger Event an. Eine Initiation Queue kann mehrere Target Queues bedienen, aber eine Target Queue ist immer einer bestimmten Initiation Queue zugeordnet.

Nehmen wir die folgende Situation an: Program A sendet eine Nachricht an ein Remote Programm B, das getriggert werden soll (siehe die beiden folgenden Abbildungen):

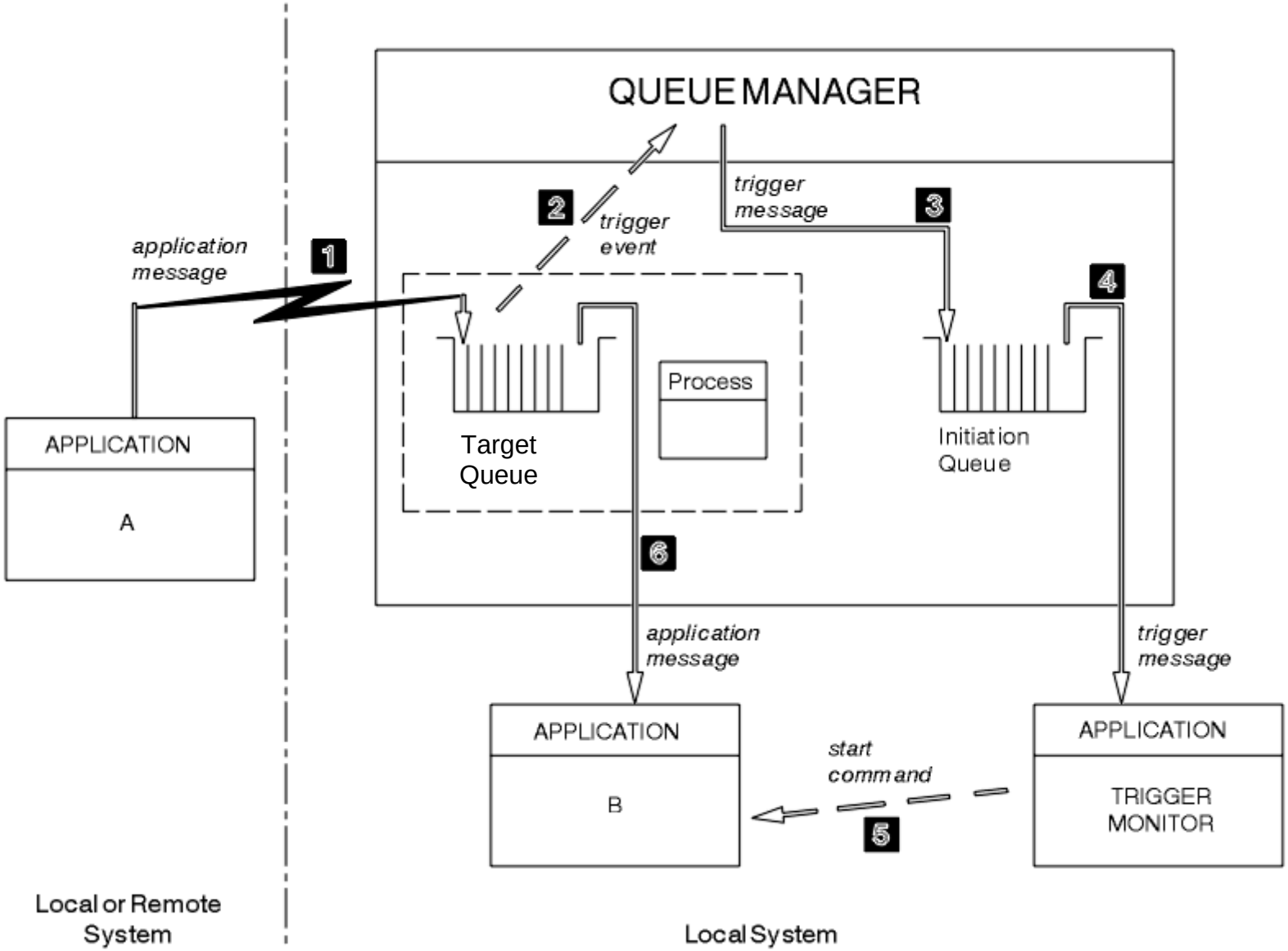


Triggering

Die Ankunft einer Nachricht in einer getriggerten Queue startet eine komplexe Folge von Ereignissen.

1. Programm A, das entfernt zu dem empfangenden Queue-Manager ist, stellt eine Nachricht in die Target Queue. Beachten Sie, dass kein Anwendungsprogramm diese Nachricht erwartet.
2. Der empfangende Queue-Manager überprüft die Nachricht. Er stellt fest, ob die Bedingungen erfüllt sind, unter denen er ein Trigger-Event erzeugen muss (u.a. Daten im Message Descriptor). Wenn ja, wird ein Trigger Event generiert.
3. Der Queue-Manager erstellt eine Trigger-Nachricht und legt sie in die Initiation Queue, welche der Target Queue zugeordnet ist. Dazu muss eine spezielle Anwendung (Trigger-Monitor) das Erscheinen der Trigger Message in der Initiation Queue zur Kenntnis nehmen..
4. Der Trigger Monitor liest die Trigger-Nachricht von der Initiation Queue.
5. Der Trigger-Monitor startet das Programm B (die Server-Anwendung).
6. Programm B öffnet die Target Queue und liest die Nachricht.

WebSphere MQ Application Programming Guide, SC34-6064-03, p.196


QUEUE MANAGER
trigger message
2
trigger event
3
application message
1
4
Process
Target Queue
Initiation Queue
APPLICATION
A
6
application message
trigger message
APPLICATION
B
start command
APPLICATION
TRIGGER MONITOR
5
Local or Remote System
Local System

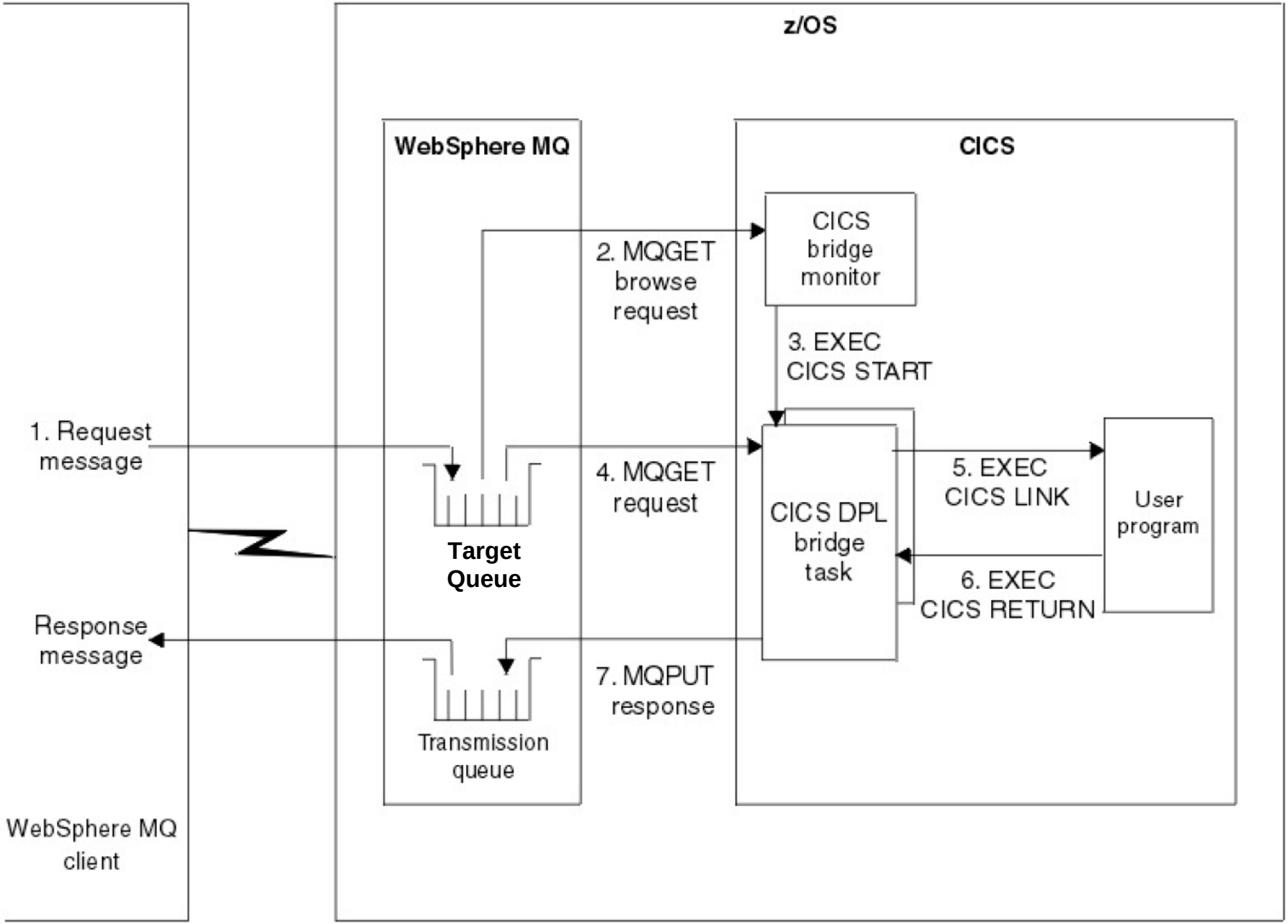
WebSphereMQ Zugriff auf eine CICS Transactions

WebSphereMQ im Request/Response-Modus ist eine Alternative zur Verwendung eines 3270-Emulator beim Zugriff auf eine CICS-Transaktion. Dies kann mit Hilfe des "MQ-CICS-Bridge" Software-Produktes durchgeführt werden. Die MQ-CICS-Bridge besteht aus 2 Komponenten, dem Bridge Monitor und der DPL Bridge Task.

Die folgende Abbildung zeigt, wie es funktioniert:

1. **Eine Nachricht, die eine CICS-Transaktion aufrufen soll, wird in der Target Queue empfangen. Die Nachricht enthält einen Trigger Event.**
2. **Der CICS Bridge Monitor übernimmt die Funktion des Trigger Monitors.**
3. **Der CICS Bridge Monitor startet eine spezielle "CICS DPL Bridge" Task.**
4. **Die CICS DPL Bridge Task liest (und entfernt) die Nachricht aus der Target Queue.**
5. **Der CICS DPL Bridge Task verwendet den Distributed Program Link (DPL)-Aufruf und greift auf eine CICS Region zu (z.B. ein Application Owning Region - AOR). Diese startet die Transaktion und führt sie aus.**
6. **Die Transaktion übergibt das Ergebnis an die CICS DPL Bridge Task, eventuell über eine COMMAREA.**
7. **Der CICS DPL Bridge Task verwendet einen MQPUT Anruf, um das Ergebnis in die WebSphere MQ Transmission Queue zu speichern.**
8. **Der Queue-Manager transportiert die Nachricht in der Transmission Queue über eine MQ-Channel an den WebSphere MQ-Client.**

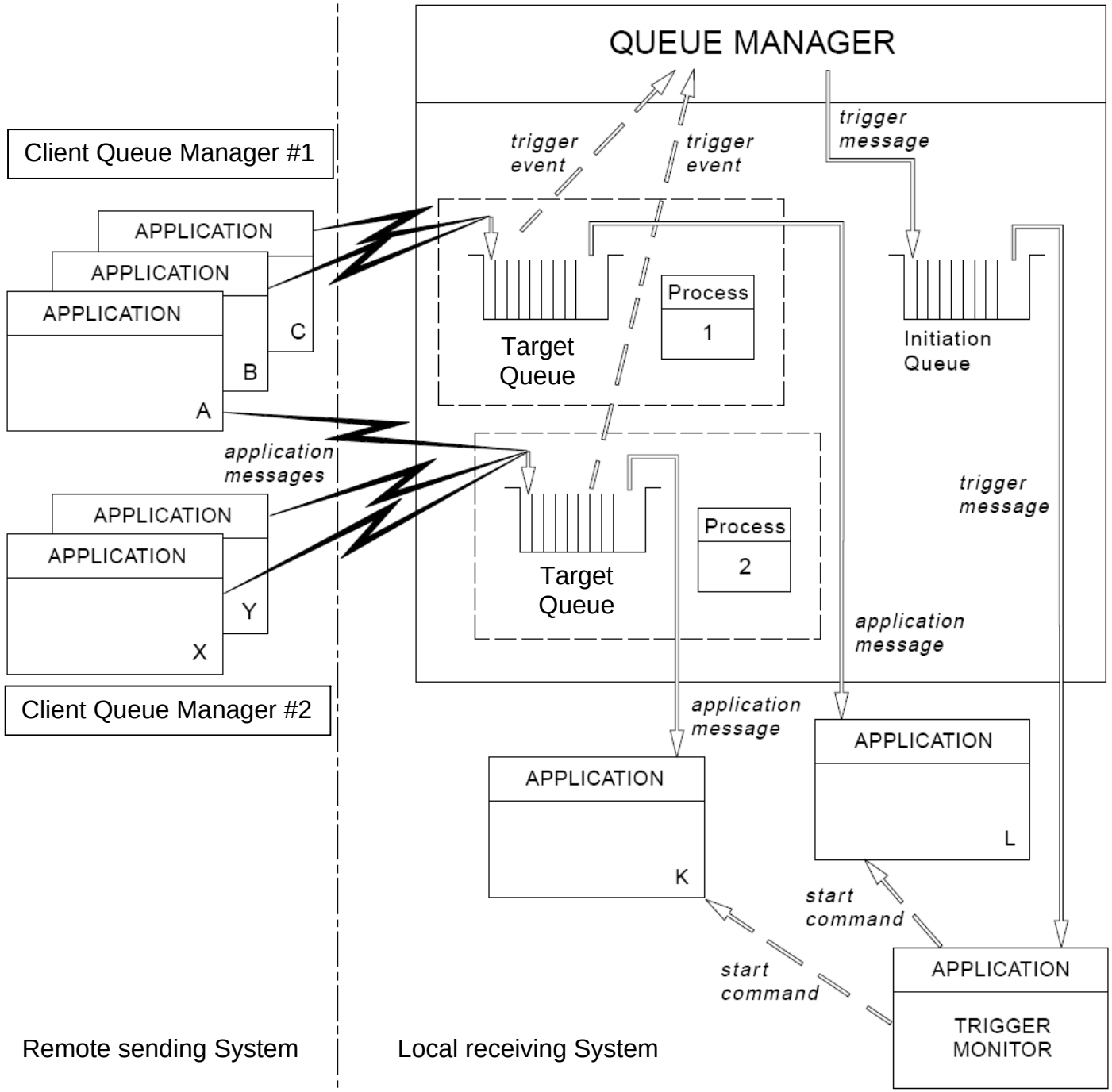
Eine detaillierte Beschreibung ist zu finden in einer Masterarbeit von Tobias Busse:
http://www-ti.informatik.uni-tuebingen.de/~spruth/DiplArb/Busse04.pdf


z/OS
WebSphere MQ
CICS
CICS bridge monitor
2. MQGET browse request
3. EXEC CICS START
1. Request message
4. MQGET request
CICS DPL bridge task
5. EXEC CICS LINK
User program
6. EXEC CICS RETURN
Target Queue
Response message
7. MQPUT response
Transmission queue
WebSphere MQ client

Multiple Target Queues

Die folgende Abbildung zeigt eine Situation, wo mehrere Programme, die entfernt zu dem empfangenden Queue-Manager sind, Nachrichten an zwei verschiedenen Target Queues senden. Mehrere Nachrichten können die gleiche Target Queue adressieren. Es ist angenommen, alle eingehenden Nachrichten zeigen Trigger-Events in ihren Deskriptoren an.

Obwohl mehrere Target Queues in Betrieb sind, und jede einem anderen Anwendungsprogramm zugeordnet ist, kann eine einzige Initiation Queue und ihr Trigger Monitor alle Triggering Events bedienen.



Gezeigt sind zwei Anwendungsprogramme K und L, die Nachrichten von verschiedenen Anwendungs-programmen A, B, C, X, Y empfangen können. Programme K und L werden von zwei verschiedenen Target Queues bedient.

Mit einer Target Queue ist ein Prozess-Definition Objekt assoziiert, welches Details über die Anwendung enthält, die die Nachricht verarbeiten soll. Der Queue Manager stellt die Informationen in eine Trigger Messager, die in einer Initiation Queue gespeichert wird.

Der Trigger Monitor extrahiert diese Informationen aus der Trigger-Nachricht und startet die entsprechende Anwendung, um die Nachricht in jeder Target Queue zu behandeln. Nur eine Initiation Queue ist erforderlich, um entweder Anwendung K oder Anwendung L zu triggern.