UNIVERSITÄT LEIPZIG

Enterprise Computing
Einführung in das Betriebssystem z/OS

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

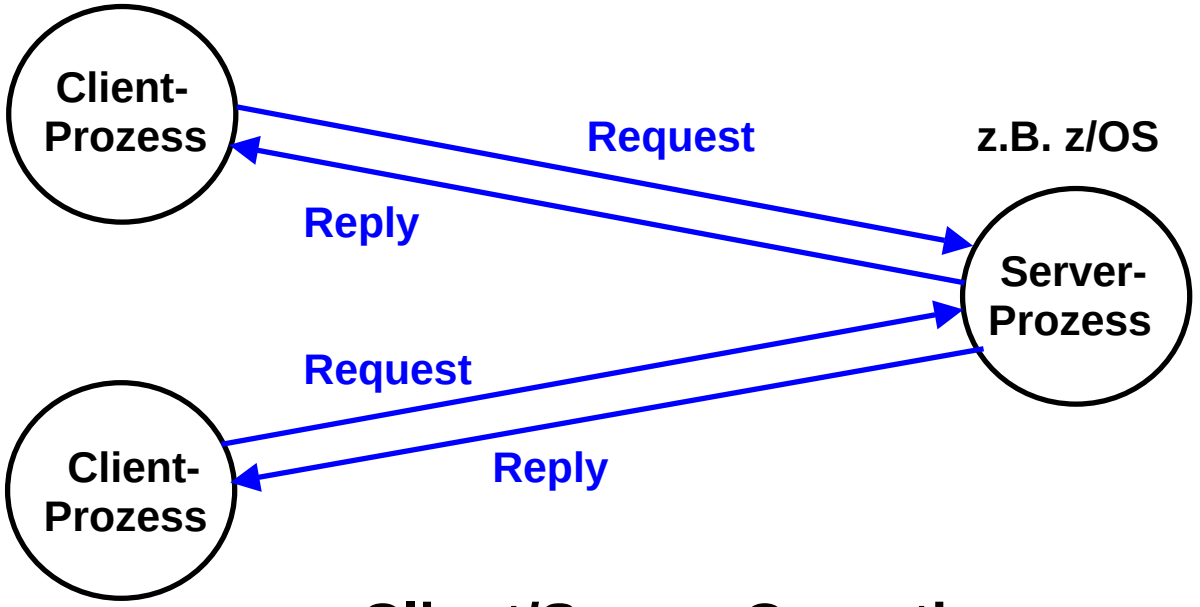
WS2012/2013

WebSphere MQ Teil 1

Übersicht

Literatur: Dieter Wackerow: MQSeries Primer. www.redbooks.ibm.com/redpapers/pdfs/redp0021.pdf
Auch hier erhältlich: http://www.informatik.uni-leipzig.de/cs/Literature/Textbooks/MQSerPrimer.pdf

Windows, Linux, Apple iOS, Android, JVM



Client/Server-Operation

Abb. 1.1

Client-Prozesse auf einem Client-System (z.B. einem PC) rufen Dienste (Services) auf einem Server-System auf. Ein Beispiel ist der Zugriff auf eine URL auf einem Apache-Webserver. Als Dienstleistung gibt Apache das gewünschte Dokument zurück. Ein Server bietet Dienste für jeden Client an, der seine Dienste anfordert.

Wir verwenden die folgenden Begriffe:

- **Client:** **ruft einen Dienst (Service) auf einem Server auf**
- **Server:** **System, auf dem Dienste ausgeführt werden**
- **Service:** **Software Prozess, der auf einem Server ausgeführt wird (möglicherweise auf mehreren Servern)**
- **Interaktion:** **Request / Reply**

Ein physischer Server bietet in der Regel viele verschiedene Dienste an.

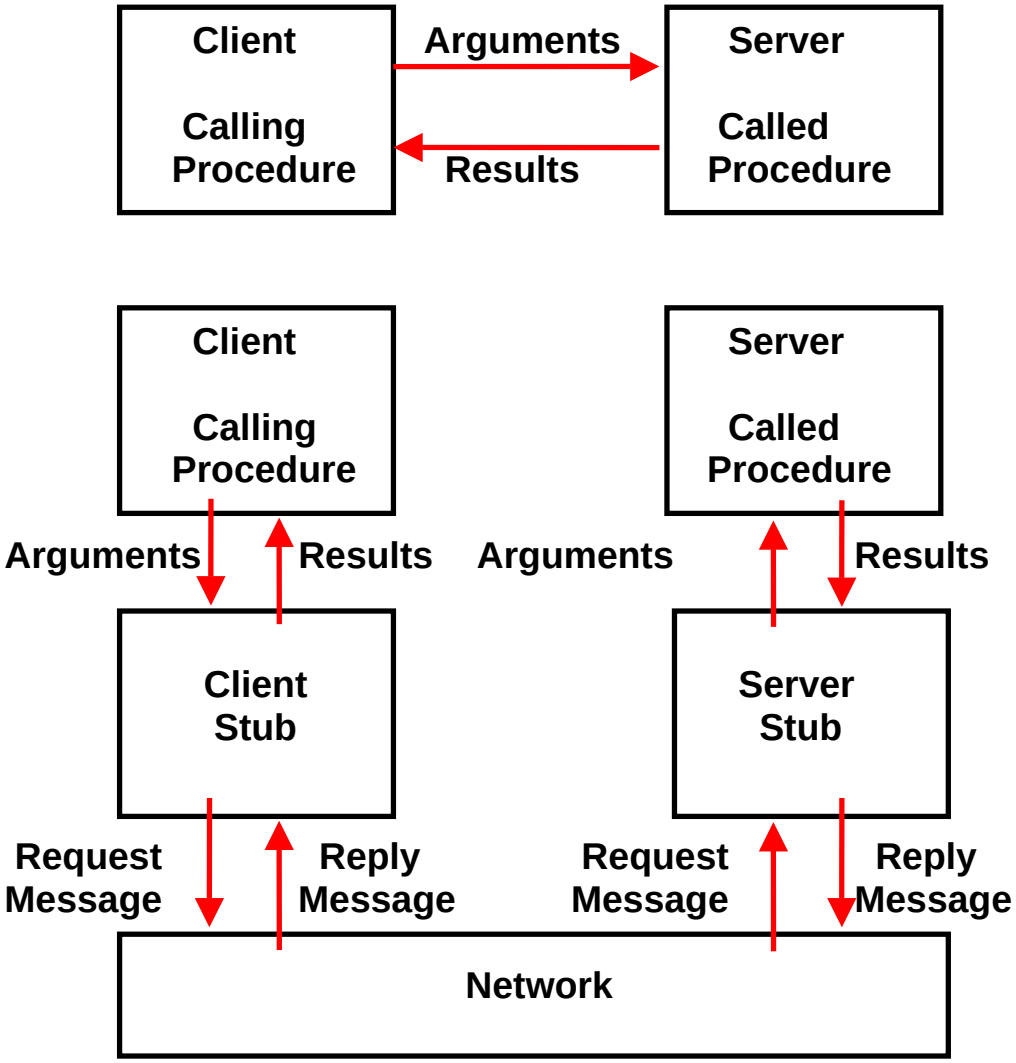


Abb. 1.2

Local Procedure Call

Eine aufgerufene Prozedur (Server) wird innerhalb des gleichen Adressraums wie die aufrufende Prozedur (Client) ausgeführt.

Remote Procedure Call

Client und Server laufen als zwei getrennte Prozesse, in der Regel auf zwei verschiedenen Rechnern. (Eine Konfiguration mit zwei getrennten Prozessen, die jeweils in ihrem eigenen virtuellen Adressraum auf dem gleichen Rechner laufen, ist ebenfalls möglich).

Beide Prozesse kommunizieren über Stubs. Stubs bilden lokale Prozeduraufrufe auf Netzwerk RPC Funktionsaufrufe ab.

Java und CORBA verwenden den Begriff Skeleton anstelle von Server-Stub.

Der Remote Procedure Call (RPC) erweckt den Anschein, als ob ein Client eine Prozedur aufruft, die sich in dem gleichen Adressraum befindet. In Wirklichkeit ruft die Client-Anwendung eine lokale Stub Prozedur auf, (anstelle des eigentlichen Codes, der die aufgerufene Prozedur implementiert). Der Client und Server haben jeweils einen eigenen Adressraum, und können sich (Normalfall) auf unterschiedlichen Maschinen befinden.

Wie funktioniert das? (1)

Stubs werden kompiliert und mit der Client-Anwendung verbunden. Statt dem eigentlichen Code, der die Remote-Verfahren implementiert, bewirkt der Client-Stub-Code:

- **Ermitteln der erforderlichen Parameter von dem Client-Adressraum.**
- **Übersetzen der Parameter in ein Standard-Format (Network Data Representation, NDR) für die Übertragung über das Netzwerk.**
- **Anruf von Funktionen in der RPC-Client-Laufzeitbibliothek, um die Anforderung und seine Parameter an den Server zu senden.**

Der Server führt die folgenden Schritte aus, um die Remote Procedure aufzurufen:

- **Die Server RPC-Laufzeitbibliothek Funktionen akzeptieren die Anfrage und rufen die Server-Stub Prozedur auf.**
- **Der Server Stub empfängt die Parameter von dem Netzwerk-Puffer und konvertiert sie von dem Netzwerk Übertragungsformat (NDR) in das Format, welches der Server benötigt.**
- **Der Server-Stub ruft die eigentliche Prozedur auf dem Server auf.**

Wie funktioniert das? (2)

Die Remote-Prozedur läuft jetzt. Sie erzeugt möglicherweise Ausgabeparameter und einen Rückgabewert. Wenn die Remote-Prozedur abgeschlossen ist, werden die resultierenden Daten in einer ähnlichen Folge von Schritten an den Client zurückgegeben:

- **Die Remote-Prozedur übergibt Ihre Ausgabeparameter und Rückgabewert Daten an den Server-Stub.**
- **Der Server-Stub wandelt die Daten auf das Format um, das für die Übertragung über das Netzwerk benötigt wird, und übergibt sie an die RPC-Laufzeitbibliothek.**
- **Die Server RPC-Laufzeitbibliothek überträgt die Daten über das Netzwerk an den Client Rechner-Computer.**

Der Client akzeptiert die Daten und übergibt sie an die aufrufende Funktion:

- **Die Client RPC-Laufzeitbibliothek erhält die Remote-Prozedur Rückgabewerte und gibt sie an den Client-Stub. weiter.**
- **Der Client-Stub wandelt die Daten aus dem NDR-Format in das Format um, das der Client-Rechner erwartet. Der Stub schreibt Daten in den Client-Speicher und liefert das Ergebnis an das aufrufende Programm auf dem Client.**
- **Die aufrufende Prozedur fährt fort, als ob sich die aufgerufene Prozedur im gleichen Adressraum befunden hätte.**

Arten von Remote Procedure Calls

Es existieren viele unterschiedliche Arten von RPCs:

Klassische RPCs

- **Sun RPC (ONC RPC)**
- **DCE RPC**

Klassische RPCs sind immer noch populär aufgrund der überlegenen Leistung. Linux, Unix und z/OS-Systeme unterstützen sowohl den Sun RPC als auch den DCE RPC.

Specialisierte RPC

- **DPL**

CICS Distributed Program Link, sehr performant, leistungsfähig und einfach zu bedienen, aber nur nutzbar zwischen CICS-Servern.

Objekt orientierte RPCs

- **Corba**
- **RMI**

Corba und RMI sind objektorientiert RPCs. RPC-Services werden als Objekte implementiert. Ein Client ruft eine Methode eines bestimmten Objekts auf.

- **Web Services RPC (SOAP RPC)**

Der Web Service RPC benutzt das Simple Object Access Protocol (SOAP).

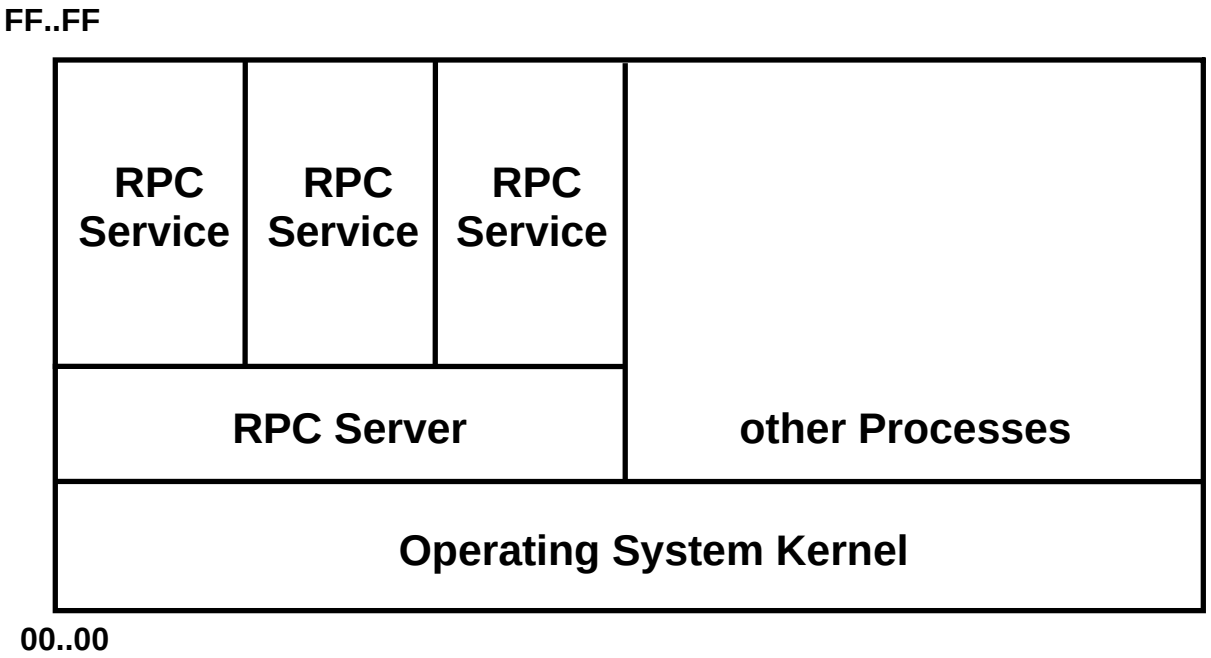


Abb. 1.3

Eine aufgerufene Prozedur wird als RPC-Service (oder Implementation) bezeichnet. Ein RPC-Server bietet in der Regel viele unterschiedliche Arten von RPC Services an. Diese Dienste können entweder in separaten virtuellen Adressräumen laufen, oder alternativ als Threads in einem einzigen Adressraum. In CICS ist die CICS Nucleus der RPC-Server; einzelne Transaktionen, die durch ihre TRID gekennzeichnet sind, sind die RPC Services.

Der RPC-Server liefert eine Laufzeitumgebung für die RPC-Services. Binding (Address Resolution) ist eine von vielen Funktionen. Bereitstellung der erforderlichen Kommunikations-Funktionen ist eine andere.

Ein Rechner kann mehrere RPC-Server beherbergen, von denen jeder eine andere Laufzeitumgebung für seine RPC-Services zur Verfügung stellt. Ein Beispiel ist ein z/OS-System, welches einen CICS Server, einen DCE-Server mit relaxten Sicherheitsmerkmalen, und einen weiteren DCE-Server mit strengen Sicherheitsanforderungen enthält.

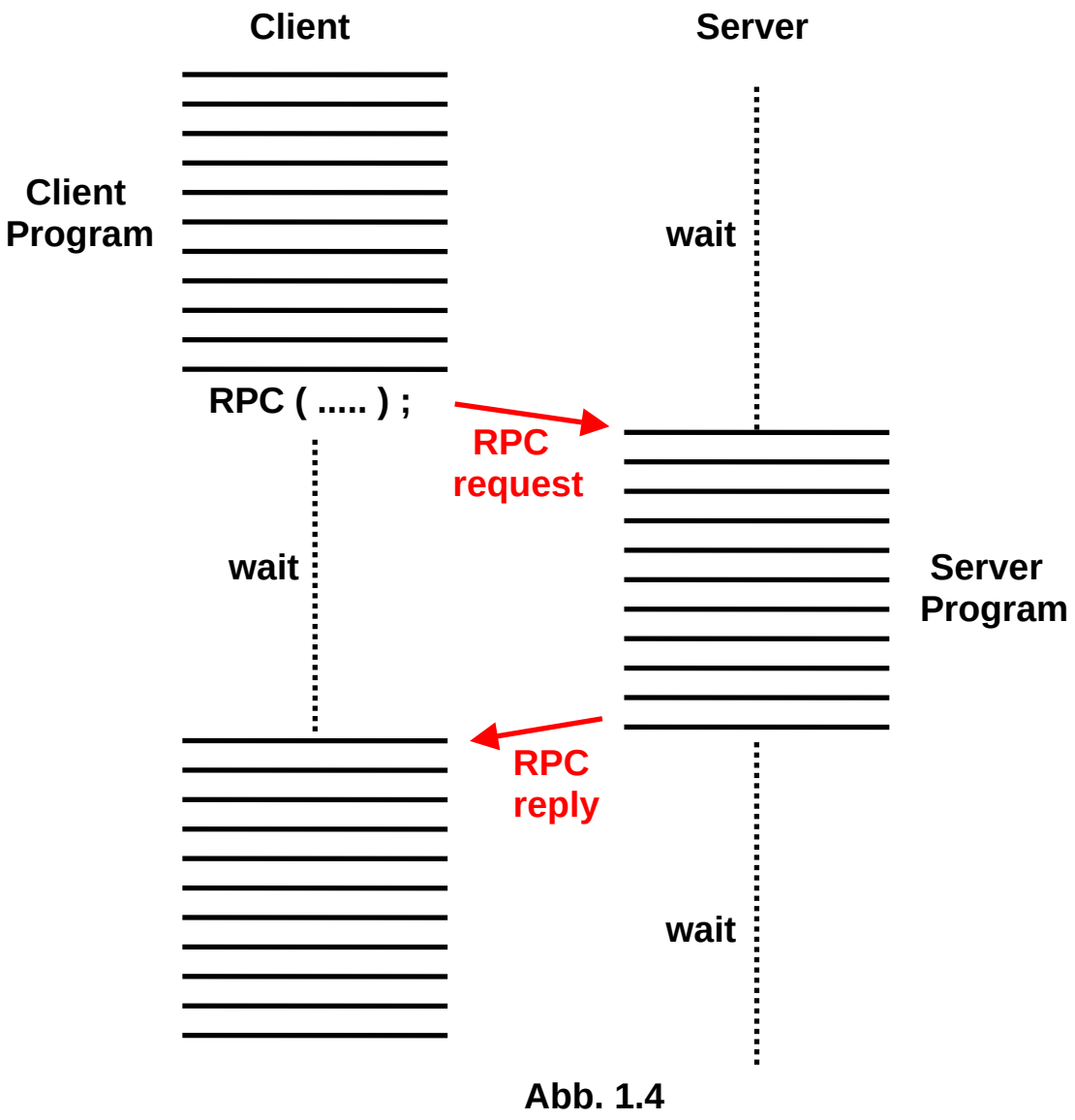


Abb. 1.4

Die meisten RPCs werden als "synchrone RPCs" implementiert. Der Client-Prozess wartet, während der Server-Prozess eine Antwort erstellt. Der Client blockiert, während der Server-Prozess ausgeführt wird.

Ein asynchrones RPC-Client-Programm wartet nicht auf eine Antwort von dem Server. Electronic Mail ist ein Beispiel für einen eher primitiven asynchronen RPC.

Synchronous RPC

Message Based Queuing (MBQ)

Auch als Message Oriented Middleware (MOM) bezeichnet

MBQ ist ein Verfahren zur asynchronen Programm zu Programm-Kommunikation. Es kann verwendet werden, um einen asynchronen RPC implementieren. Programme bekommen die Fähigkeit, dass sie Informationen senden und empfangen können, ohne dass eine direkte Verbindung zwischen ihnen besteht. Programme kommunizieren, indem sie Nachrichten in Message-Queues hinterlegen, und Nachrichten von Message-Queues abrufen.

Führende MBQ Produkte sind:

- **IBM MQSeries, jetzt umbenannt in WebSphere MQ**
- **Microsoft MS Message Queue Server (MSMQ)**
- **Oracle BEA MessageQ**

MQSeries ist seit 1993 verfügbar und ist der unangefochtene Marktführer. Das Produkt ist für mehr als 35 Betriebssystem-Plattformen verfügbar, einschließlich AIX, DG / UX, HP-UX, i5/OS, Sequent, Sinix, Solaris, Tandem, TPF, TrueUnix, VMS / VAX, Windows und z/OS.

Anders als bei anderen MBQ Produkte ist Microsoft MSMQ nur auf Windows-Plattformen verfügbar. Es biete auch keine native Unterstützung für JMS (Java Messaging Service) an.

Apache ActiveMQ ist das populärste Open-Source MBQ Produkt.

Message Based Queuing ist attraktiv für die Integration von heterogenen Hardware, Software und Anwendungsumgebungen.

MBQ Funktion

E-Mail wie SMTP, das Simple Mail Transport Protocol, und X.500 sind primitive MBQ Anwendungen. Merkmale sind:

- **Eine Nachricht wird gesendet.**
- **Die Nachricht kann (oder auch nicht) am Ziel ankommen.**
- **Der Empfänger ist (in der Regel) ein Mensch, der beschließt, die Mail, sofort oder später oder auch gar nicht zu beantworten.**

MBQ unterscheidet sich von elektronischer Post:

- **MBQ bietet ACID Eigenschaften, die sicherstellen, dass eine Nachricht genau einmal ausgeliefert wird.**
- **Ein MBQ Nachricht wird durch ein Anwendungsprogramm empfangen, das weiß, was damit zu tun ist.**

Message Queuing Eigenschaften

Message Queuing ist eine Methode der Programm-zu-Programm-Kommunikation. Programme innerhalb einer Anwendung kommunizieren miteinander durch das Schreiben und Abrufen von anwendungsspezifischen Daten (Nachrichten) zu/von Warteschlangen (Queues), ohne über eine private, dedizierte logische Verbindung miteinander verknüpft zu sein. Messaging bedeutet, dass Programme miteinander durch Senden von Daten in Nachrichten kommunizieren und nicht durch einen direkt Aufruf.

Queuing bedeutet, dass Programme über Nachrichten kommunizieren, die in Queues gespeichert werden. Anders als bei einem synchronen RPC, müssen Programme, die über Queues kommunizieren, nicht gleichzeitig ausgeführt werden. Ein Java-Objekt kann eine Methode eines anderen Java-Objekts durch Senden einer Nachricht aufrufen. Dies ist jedoch eine synchronen und keine asynchrone Kommunikation.

Beim asynchronen Messaging fährt das sendende Programm mit seiner Verarbeitung fort, ohne auf eine Antwort seiner Nachricht zu warten. Im Gegensatz dazu wartet ein synchrones RPC Programm auf eine Antwort, ehe es die Verarbeitung fortgesetzt.

Beim Message Queuing muss sich der Programmierer nicht darum kümmern, ob das Zielprogramm beschäftigt oder nicht verfügbar ist. Er macht sich noch nicht einmal Gedanken, ob der Server heruntergefahren oder die Verbindung zum Server gestört ist. Der Programmierer sendet Nachrichten an eine entfernte (remote) Queue, die einem Anwendungsprogramm zugeordnet ist. Letzteres kann zu diesem Zeitpunkt verfügbar sein oder auch nicht. WebSphere MQ kümmert sich um den Transport zu der Zielanwendung. Ggf. startet es diese sogar.

Für den Anwender ist das zugrunde liegende Kommunikationsprotokoll transparent.

WebSphere MQ

Das IBM MQSeries Software-Produkt ist seit langer Zeit verfügbar. Kürzlich hat IBM entschieden, MQSeries in WebSphere MQ umzubenennen. Sowohl die alte Bezeichnung MQSeries, der neue Begriff WebSphere MQ und die Kurzform MQ werden mehr oder weniger synonym verwendet.

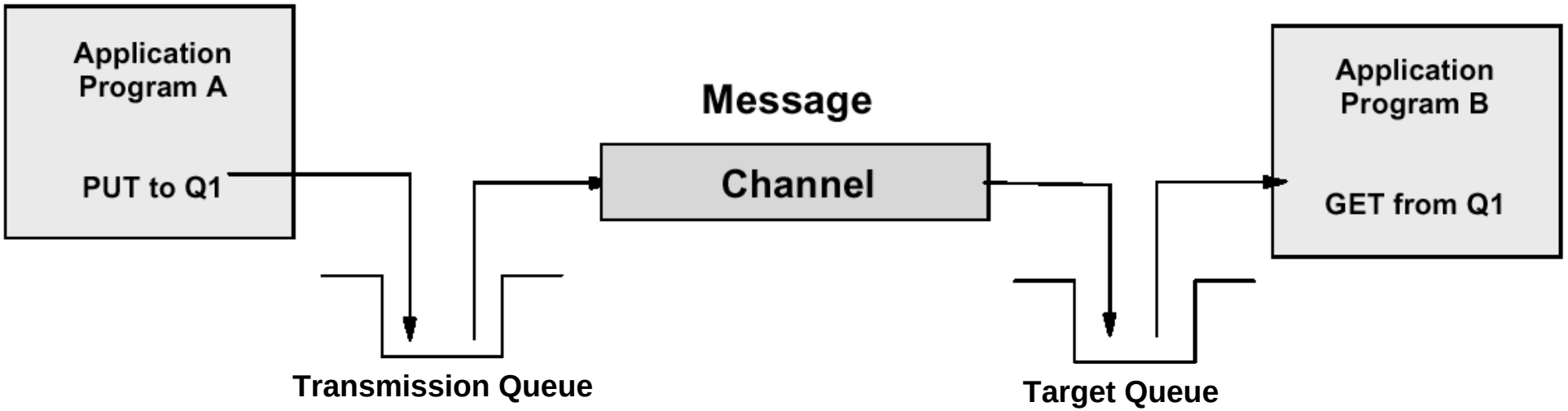
IBM WebSphere entstand als IBMs Java Application Server-Produkt, heute bekannt unter dem Namen WebSphere Application Server (WAS). IBM hat beschlossen, den Namen WebSphere wiederzuverwenden, um eine Reihe von mehr oder weniger verwandten Software-Produkten (einschließlich WAS) zu kennzeichnen. Neben dem WebSphere Application Server enthält die WebSphere-Familie Produkte wie den WebSphere Designer, den WebSphere Integration Developer, WebSphere Process Server oder WebSphere Portal Server. Es gibt über 200 Software-Produkte in der WebSphere-Familie. Einige der Produkte sind gut miteinander integriert, andere nicht. WebSphere MQ ist ein ziemlich eigenständiges Add-on, das unabhängig von dem WebSphere Java Application Server eingesetzt werden kann.

Die meisten Personen, wenn sie den Namen WebSphere benutzen, meinen damit den WebSphere Application Server (WAS).

Es gibt 2 Versionen von WebSphere MQ:

- **Integrierte Version,**
- **Unabhängige (nicht integrierte) Version, die nur wenige Beziehungen mit dem Rest der WebSphere-Produktfamilie hat.**

Viele Personen benutzen immer noch den Namen MQSeries für die unabhängige (nicht integrierte) Version von WebSphere MQ. Der WebSphere Application Server (WAS) Implementierung des Java Message Service (JMS)-Standards verwendet hierfür eine integrierte Version von WebSphere MQ.



Program-to-Program Communication

Ein Anwendungsprogramm kann Nachrichten an einen anderen Anwendungsprogramm senden, das auf einem entfernten System, wie etwa einem Server oder einem Host läuft.

Der sendende Anwendungsprogramm A speichert die Nachricht in eine lokale Transmission Queue mit dem MQPUT Befehl . Von dort aus wird die Nachricht über eine "Message Channel" zu einer entfernten Target Queue übertragen. Das empfangende Anwendungs-Programm B kann nun die Nachricht aus der Target Queue mit dem MQGET Befehl zu einem Zeitpunkt seiner Wahl abrufen.

Die "Message Channel" (oder Channel) ist ein MQSeries Konstrukt, das benutzt wird, um Nachrichten von einer Transmission Queue in eine Target Queue zu übertragen. Es ist ein Schicht 5 Protokoll, vergleichbar mit Telnet, 3270, http oder SMTP. Es benutzt TCP/IP (oder SNA) als den darunter liegenden Schicht 4 Transportmechanismus.

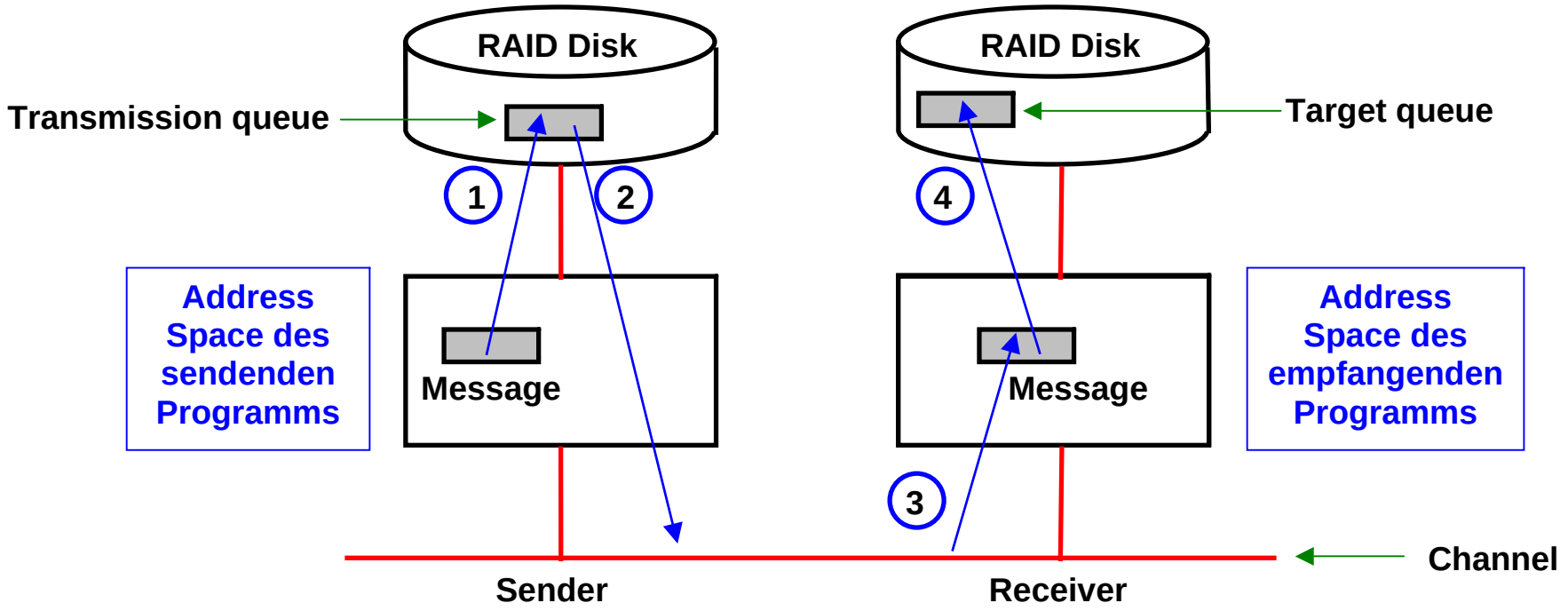


Abb. 1.5

Persistente und Non-Persistente Nachrichten

WebSphere MQ unterscheidet zwischen persistenten und nicht-persistente Nachrichten. Die Übertragung von persistenten Nachrichten ist gewährleistet, und überlebt Systemausfälle, ähnlich wie Nachrichten, die CICS in seine Log Files schreibt. Nicht-persistente Nachrichten können nach einem Systemausfall nicht wiederhergestellt werden.

Die Persistenz wird mit Hilfe des Speicherns der Nachricht in einem lokalen RAID Festplatten-Subsystem erreicht.

Persistente Nachrichten implementieren eine Kommunikation mit ACID-Eigenschaften (die Zustellung der Nachricht genau einmal wird garantiert). Die Nachricht wird dauerhaft (persistent) durch den Absender gespeichert, ehe sie gesendet wird. Der Empfänger speichert die Nachricht ebenfalls persistent, ehe sie verwendet wird, und bestätigt den Empfang an den Absender.

Persistent Messaging Operation

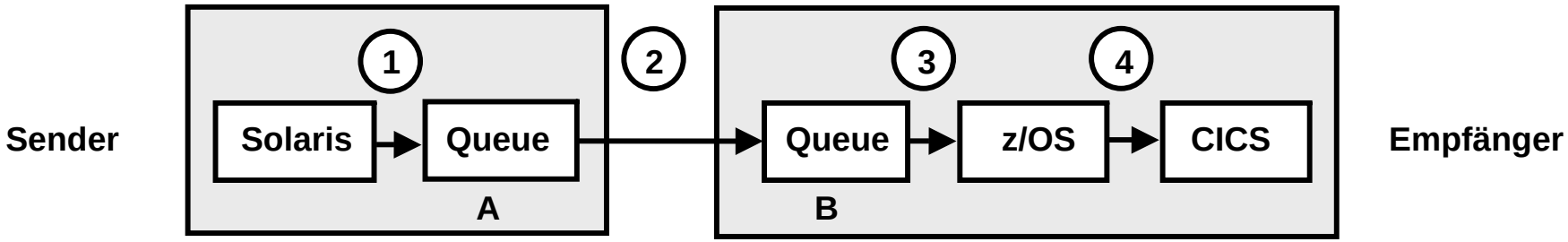


Abb. 1.6

Die persistente Nachrichtenübertragung erfolgt in mehreren Schritten.

Schritt 1: Der Sender speichert die Nachricht persistent in der lokalen Transmision Queue A.

Schritte 2 - 3: Die Nachricht wird über das Netzwerk an die Target Queue B übertragen. Wenn die Übertragung erfolgreich war, sendet der Empfänger eine Bestätigung. Wenn erfolglos, wird der Vorgang beliebig oft wiederholt.

Schritt 4: Der empfangende Anwendungsprogramm (CICS in diesem Beispiel) kann nun die Nachricht aus Queue B zu einem Zeitpunkt seiner Wahl abrufen. Dieser Zeitpunkt kann beliebig weit in der Zukunft liegen.

Wenn ein Anwendungsprogramm eine Nachricht in einer Queue speichert, gewährleistet MQSeries, dass die Nachricht:

- **Sicher gespeichert wird,**
- **recoverable ist, und**
- **nur einmal, und genau einmal (exactly once) an die empfangende Anwendung ausgeliefert wird.**

Die Kommunikation ist einseitig.

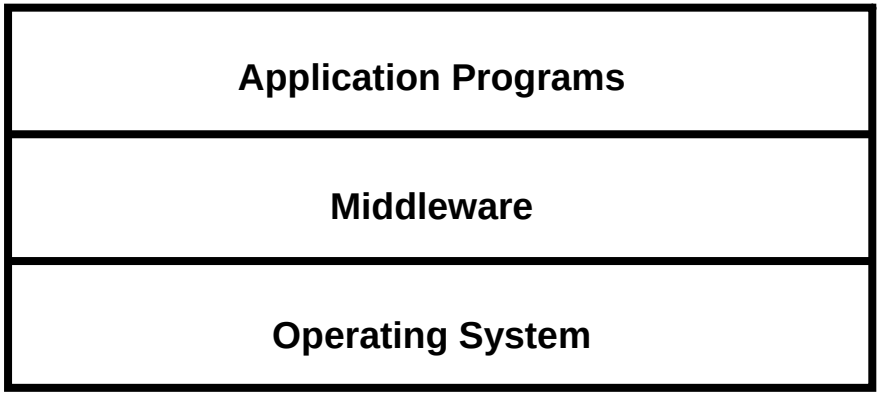


Abb. 1.7

Middleware

Der CICS Nucleus ist eine Runtime-Ausführungsumgebung, welche die gleichzeitige Ausführung von mehreren unabhängigen Anwendungsprogrammen (Transaktionen) ermöglicht. Ähnliche ermöglicht ein RPC-Server die gleichzeitige Ausführung von mehreren unabhängigen Anwendungsprogrammen. Der WebSphere Application Server (WAS) ist eine Laufzeit-Ausführungsumgebung, die die gleichzeitige Ausführung von mehreren unabhängigen Java-Anwendungsprogrammen ermöglicht, unter Benutzung von Einrichtungen wie einem Java Servlet-Container und einem Enterprise Java Bean (EJB) Container.

Der Queue Manager ist eine Runtime-Ausführungsumgebung, die die gleichzeitige Ausführung von mehreren unabhängigen WebSphere MQ Anwendungsprogrammen ermöglicht.

Software wie der CICS Nucleus, der RPC-Server, der Web Application Server und der Queue Manager wird gemeinhin als Middleware bezeichnet. Sie alle stellen eine gemeinsame Laufzeitumgebung für mehrere unabhängige Anwendungsprogramme zur Verfügung.

Queue Manager

Der Kern von WebSphere MQ ist der (Message) Queue-Manager (MQM), ein WebSphere MQ Run-Time Programm.

Ehe ein Anwendungsprogramm WebSphere MQ auf einem Rechner verwendet, muss ein Queue Manager gestartet werden. Der Queue-Manager besitzt und verwaltet die Ressourcen, die von WebSphere MQ verwendet werden. Zu diesen Ressourcen gehören:

- **Page Sets, die WebSphere MQ Objektdefinitionen und Message Daten enthalten,**
- **Log Files, die verwendet werden, um Nachrichten und Objekte wiederherzustellen falls ein Queue Manager Fehler auftritt,**
- **Prozessor Speicherplatz,**
- **Anschlüsse, über die verschiedenen Anwendungsumgebungen (zB CICS, IMS, Batch) auf die WebSphere MQ API zugreifen können,**
- **Den WebSphere MQ Channel-Initiator (später besprochen), der die Kommunikation zwischen WebSphere MQ auf einem sendenden und einem empfangenden Rechner ermöglicht**

Der Queue-Manager hat einen Namen, und Anwendungen können sich mit einem entfernten Queue Manager über seinen Namen verbinden.

Die Aufgabe des Queue Managers ist es, Warteschlangen und Nachrichten für Anwendungen zu verwalten. Es stellt eine API, die Message Queuing Interface (MQI), zur Kommunikation mit Anwendungen zur Verfügung. Anwendungs-programme benutzen Funktionen des Queue Managers über API-Aufrufe (Commands). Zum Beispiel legt der MQPUT API-Aufruf eine Nachricht in eine Queue, die durch einentferntes Programm über den API-Aufruf MQGET gelesen werden kann. Dieses Szenario ist auf der nächsten Seite gezeigt.

Der Queue Manager kann Nachrichten für den Fall von Anwendung- oder Systemausfällen zwischenspeichern. Nachrichten werden in einer Warteschlange gespeichert, bis eine erfolgreiche Antwort durch die MQI empfangen wird.

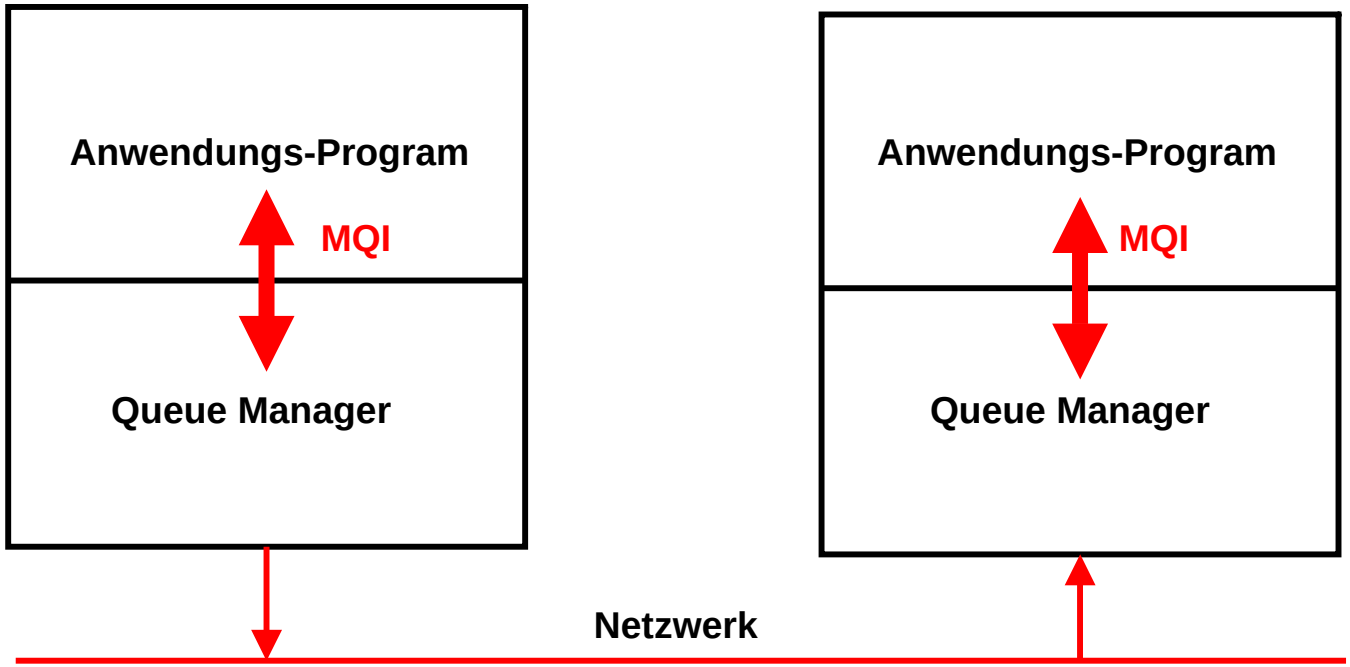


Abb. 1.8

WebSphere MQ ermöglicht es Anwendungsprogrammen, miteinander über ein Netzwerk ungleicher Komponenten, wie unterschiedlicher Prozessoren, Subsysteme, Betriebssysteme, Kommunikationsprotokolle und Programmen, die in unterschiedlichen Programmiersprachen geschrieben sind, zu kommunizieren.

Die Abbildung zeigt die wichtigsten Teile eines WebSphere MQ-Anwendung zur Laufzeit. Anwendungsprogramme benutzen WebSphere MQ-API-Aufrufe, die Message Queue Interface (MQI), um mit einem Queue Manager zu kommunizieren. Die MQI ist eine konsistente Application Program Interface (API) für viele, sehr unterschiedliche Plattformen. Der Queue Manager ist die Laufzeit-Umgebung (runtime environment) von WebSphere MQ.

Anwendungsprogramme können in verschiedenen Programmiersprachen geschrieben werden, einschließlich Java. Der gleiche Queuing-Mechanismus gilt für alle Plattformen. Dies gilt auch für die derzeitig 13 API-Aufrufe der MQI.

Message Queuing Interface (MQI)

Die wichtigsten MQI Befehle sind:

MQCONN	**Verbinden (Connect) mit einem (in der Regel entfernten) Queue Manager**
MQOPEN	**Öffnen (Open) einer spezifischen Queue**
MQPUT	**Eine Message in eine Transmission Queue stellen**
MQGET	**Eine Message aus einer Target Queue auslesen**
MQCLOSE	**Schließen (Close) einer Queue**
MQDISC	**Verbindung zu einem Queue Manager auflösen (Disconnect)**

Neben der MQI können mehrere andere APIs benutzt werden. Die beiden wichtigsten APIs sind:

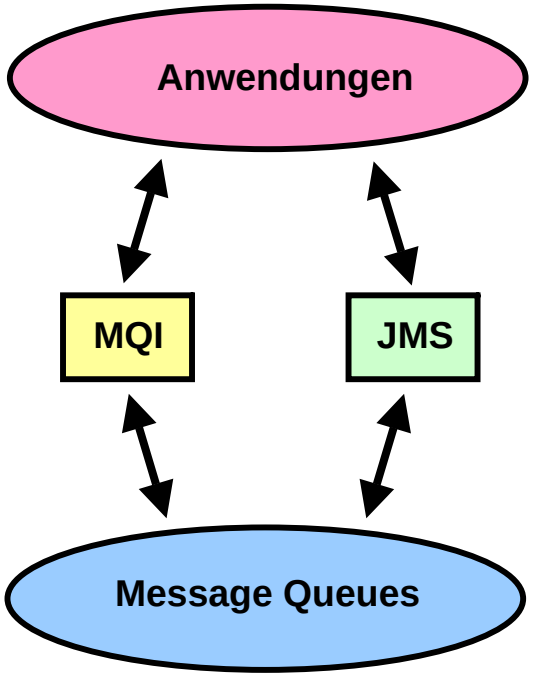


Abb. 1.9

Anwendungen greifen auf die WebSphere MQ Services über ein Application Programming Interface zu.

Die älteste und immer noch beliebteste API ist die MQI.

JMS wird von Java-Anwendungen verwendet.

Message Queue Interface (MQI)

MQI Funktionen sind verfügbar in den folgenden Sprachen:
z/OS Assembler, C/C++, COBOL, Lotus Script, Java, PL/1, VisualBasic, C#, viele andere.

Java Message Service (JMS)

Java Standard, der von WebSphere MQ unterstützt wird
Abstracts WebSphereMQ Details
Eine von mehreren Schnittstellen für JEE / Enterprise Java Beans.

Queue Transmission

Die zu übertragende Nachricht wird

1. **zuerst in einer Transmission Queue gespeichert, die lokal zu dem Anwendungsprogramm ist, das die Nachricht sendet. Von dort aus wird sie**
2. **über das Netzwerk an ein Target Queue transportiert**
3. **Diese ist lokal (auf dem gleichen Rechner) zu dem Anwendungsprogramm, welches die Nachricht empfangen soll.**

Damit dies funktioniert, ist es notwendig, dass ein Queue Manager auf dem sendenden Rechner, und ein anderer Queue Manager auf dem empfangenden Rechner existiert.

Bitte denken Sie daran, WebSphere MQ ist eine unidirektionale Kommunikation. Um einem Request / Response-Modus zu implementieren, muss eine zweite unidirektionale Verbindung vom MQ-Server zu dem MQ-Client vorhanden sein.

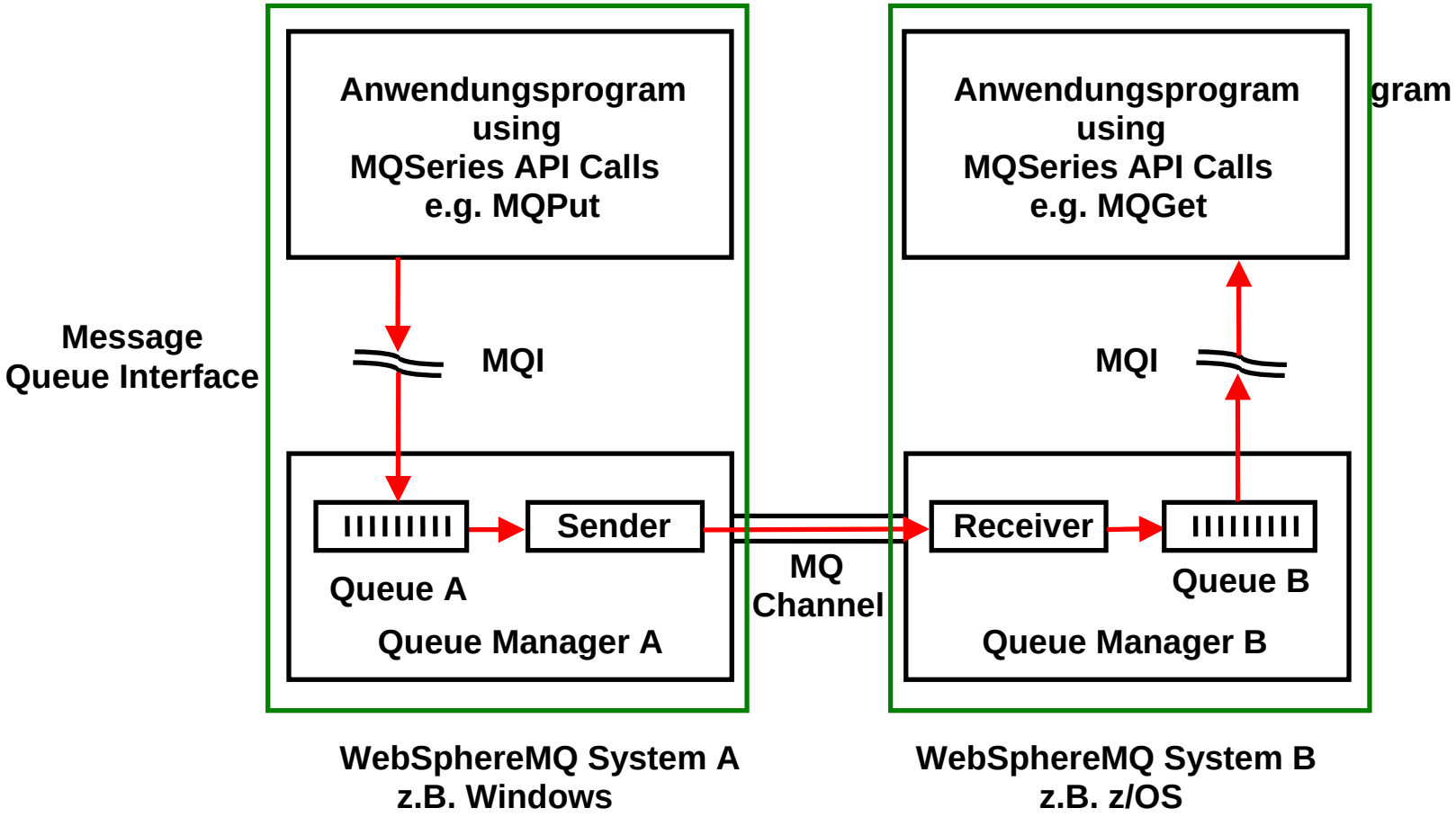


Abb. 1.10

WebSphere MQ Operation

Das Anwendungsprogramm stellt eine Nachricht in die Queue A (Transmission Queue), mit Hilfe des MQPUT Calls, ein Bestandteil der Message Queue Interface (MQI). Queue A ist Bestandteil von Queue Manager A. Dieser benutzt sie, um die Nachricht über den MQ-Channel an den Empfänger von Queue Manager B zu senden. Queue Manager B besitzt Queue B (Target Queue, auch als Request Queue, Application Queue, oder Destination Queue bezeichnet), wo die Nachricht gespeichert wird. Das empfangende Anwendungsprogramm kann die Nachricht aus der Target Queue B zu einem von ihm gewählten Zeitpunkt abrufen, mit Hilfe der MQI, beispielsweise mit dem MQI Befehl MQGET.

Sneakernet

Die Infrastruktur eines großen Unternehmens besteht neben dem Mainframe aus einer Vielzahl unterschiedlicher Rechner mit unterschiedlichen Betriebssystemen. Sehr häufig dienen die Ergebnisse auf einem Rechner als Input für den nächsten Rechner.

In der Vergangenheit war es üblich, die Ergebnisse eines Rechners in der Form von Magnetbändern oder Lochkarten zu Fuß zum nächsten Rechner zu tragen. Dies geschah durch einen menschlichen Operator, daher der Ausdruck „Sneakernet“. Noch in den 90er Jahren wurden jeden Abend umfangreiche Daten, die im Werk Bremen der Firma Daimler Benz anfielen, in der Form von Magnetbändern in einen PKW geladen, um am nächsten Morgen im Werk Sindelfingen verarbeitet zu werden.

Andrew S. Tanenbaum: „Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway.“

WebSphere MQ ist ein perfekter Ersatz für das Sneakernet: Einheitliche Schnittstellen, Sicherheit, sowie asynchrones Verhalten.

Unterstützung einer Service Oriented Architecture

WebSphere MQ ist eine Schlüsselkomponente bei der Umsetzung einer Enterprise Application Integration (EAI) oder einer Service-Oriented Architecture (SOA)-Strategie, in dem es das Messaging-Backbone für über 80 verschiedene Platformen bereitstellt. Die wachsende Bedeutung von EAI-und SOA und das Wachstum von Web Services und anderen Verbindungs-Mechanismen sind eindeutig wichtige Entwicklungen.