Java™ Message Queue

Router Deployment Topologies White Paper

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
1 (800) 786.7638
1.512.434.1511

Please
Recycle

Adobe PostScript™

CHAPTER 1

Introduction

This document presumes a basic familiarity with messaging systems, and more specifically, the Java™ Message Service (JMS) API. Systems that use Java™ Message Queue (JMQ) technology conform to a very simple, yet elegant architecture. Any number of applications can connect to a JMQ technology-enabled router. Applications may send messages (message producers), receive messages (message consumers), or both. The JMQ router is responsible for routing and delivering messages to those applications that are interested in them. Figure 1 illustrates the relationship of JMQ routers and applications.

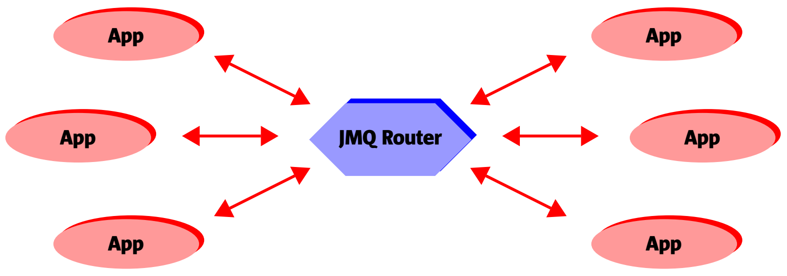

FIGURE 1 Routers and Applications

The JMQ architecture is very powerful, enabling routers to decouple message producers from message consumers. This means that an application sending a message does not need to know who will be receiving the message or how many applications might need to receive the message. Conversely, the message consumers do not need to know where a particular message originated. The overall topology of distributed systems based on JMQ technology is simplified because applications only need a single connection to the router rather than multiple connections to all the applications that they interact with.

The router also allows consumer and producer applications to be time decoupled. A producer could publish messages to a queue or topic and then terminate. Some unspecified period of time later, a consumer could start up, receive the posted messages, and process them appropriately. This time-decoupled distributed system paradigm can be a powerful way of integrating disparate applications, including legacy systems.

With JMQ technology, you can run multiple router processes and interconnect them in a wide variety of topologies. Applications connected to a JMQ router transparently publish and receive messages to and from any other application regardless of which router the other application is connected to (providing that a path exists from one application to the other through interconnected routers).

JMQ routers also support the concept of subnets. If an organization has multiple distributed systems — based on JMQ technology — that need to be isolated from each other, each distinct system (a collection of routers and applications) can run on a different subnet. Applications and routers on different subnets are completely isolated from each other.

JMQ routers are very similar to hardware routers that move TCP/IP packets across a real network. JMQ software and hardware routers transparently route messages from node to node till they reach their destination. However, JMQ routers offer time-independent message store/forward (queuing) functions and the ability to dynamically change message network topologies. It is not inconceivable that hardware-based appliances offering dedicated JMQ router services will appear in the near future.

This white paper discusses the various JMQ router interconnection topologies with respect to performance, reliability, and usability issues.

CHAPTER 2

Overview of Java Message Queue Router Operation

JMQ technology-enabled routers perform a number of key functions:

- **Connection Management** – Routers are responsible for handling application connections, including detection of new connections, termination of connections, and loss of connection due to abnormal circumstances (network or application failure). Routers handle connections to adjacent routers in a similar manner. In the event of problems, reconnection, retry, and rerouting using alternate paths are also handled by routers.
- **Message Routing** – All message routing operations go through routers. They keep track of which consumer applications are interested in what messages, and ensure that only the messages a consumer cares about are sent to that application. This also entails keeping track of which routers are interconnected and forwarding messages to the next router on the way to the final destination.
- **Message Store** – If a message needs to be persistently stored (in a queue or topic), the routers ensure that this occurs. In that regard, they manage a back-end persistent storage mechanism for messages. If applications use durable subscriptions, routers also keep track of information about the durable subscription between invocations of an application, for example, what messages were already received and any new messages that might have been published between application executions.
- **Interest Propagation** – Message consumers typically create a subscriber or receiver with a specific message selector that specifies the messages that they are interested in receiving. Message selectors use an SQL-like syntax to specify what properties need to be matched in a message header. Routers keep track of this interest information and also propagate the interests between themselves. The interest information is much like routing tables in hardware routers, albeit more complex.

CHAPTER 3

Java Message Queue Router Interconnection

When it comes to JMQ router interconnection topologies and performance tuning, there is no simple answer as to the best topology for a given situation. Each application, system, and organization is different, with unique infrastructure, design, and requirements to consider. This paper outlines considerations, options, and “rules of thumb” that are useful in evaluating JMQ technology-enabled router topologies.

Performance and tuning factors that should be considered include message latency, message throughput, and data throughput. These three factors are interrelated, so improving one might make another worse. All three are also related to message size.

- **Message Latency** – The average time that it takes to send a message from one application to another. Message latency is affected by the number of hops or routers that a message has to pass through from source to destination, and also by the speed of the underlying transport medium. Each router that a message passes through adds fixed overhead to the latency.
- **Message Throughput** – The average number of messages sent per second. Message throughput typically remains constant until the message size (in total, including headers, properties, and body/payload) exceeds one Kbyte, at which time it drops off.
- **Data Throughput** – The average number of bytes sent per second. Data throughput rises relative to the message size (more data is transmitted for larger messages relative to the overhead) and peaks at a message size of 10 Kbytes on most hardware platforms.

To optimize throughput in the correct area, it is important to evaluate the requirements of the JMQ technology-enabled application that is being deployed. If large quantities of data (for example, streaming video) need to be transmitted, data throughput should be emphasized. Latency is not that critical once a steady state is achieved. If application responsiveness is based on multiple small messages being transmitted (for example, in a near real-time, process control application), then message throughput and latency are more important.

A router propagates interest information and connection information for all applications attached to it. This overhead can affect overall performance and throughput rates if the number of connections per router grows too large. When using the Java Message Service (JMS) API, each connection goes from the application to the router. For smaller machines such as PCs, connections per router should be kept below 100 by subdividing the JMQ router network using one of the topologies discussed in this white paper. This introduces more message latency due to the extra hop but also reduces router overhead, potentially increasing message and data throughput. Large servers such as Sun™ systems running the Solaris™ Operating Environment can typically support many more concurrent connections than desktop PCs.

Common router topologies include:

- **Simple** – The use of only one or two routers
- **Linear** – Routers are linked in a straight line
- **Interconnected** – Routers are linked in a network like a spider's web
- **Hierarchical** – Routers are connected in a pyramid or hierarchy
- **Gateway** – Different topologies are linked by a gateway between two routers
- **Mixed** – Combines some or all of the above into more complex topologies

Subsequent sections discuss each of these topologies in more detail.

Simple Topologies

Simple router topologies use one or two routers, and this is adequate for many systems. In fact, JMQ technology-enabled routers are optimized to handle cases just like this, where interest and connection information does not need to be widely broadcast across the router network. Provided that the number of connections does not grow too large (beyond 100 connections per router), simple topology provides good message and data throughput with minimal latency.

However, in this scenario, the entire system is dependent on the availability of the router. If the router goes down unexpectedly due to hardware failure, operator error, or other reasons, the whole application goes down —– or half of it, in the case of two routers. For applications with high availability requirements, this may not be an acceptable solution.

Furthermore, if the system uses a large number of message types coupled with high message volumes, but interest in the messages is rather sparse and distributed over many different and distinct producer and consumer applications, the router could get overwhelmed with the message volume (depending on the horsepower available on the router's hardware platform). In that situation, it may be better to partition the producer and consumer applications by message type and use the more flexible topologies mentioned later in this paper. This enables producers and consumers of similar high-volume message types to share a router, so these messages are not propagated outside that local router. Only globally common messages that cross these logical functional boundaries will propagate to other routers and applications. Message traffic is partitioned logically across multiple routers with minimal throughput or latency impact on the high-volume messages.

Simple topologies are commonly used during the development and testing processes, when each developer runs their applications and router on a single machine and a unique subnet to isolate it. Simple topologies are also chosen for smaller departmental applications that run on a single LAN segment. Administration is kept to a minimum with a maximum of two routers, and fault-tolerance is not a key requirement in such installations.

Linear Topologies

A linear router topology puts all the routers in a straight line, each connected to two neighbors except for the end routers which only have one interconnect. A diagram of a typical linear topology is shown in Figure 1.

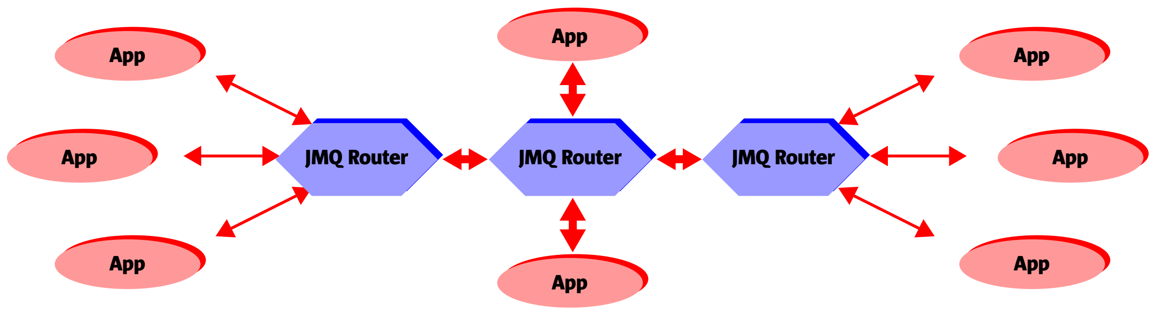

FIGURE 1 Linear Router Topology

This type of topology, especially with only three routers, can be useful when two distinct applications (one on the left and a different one on the right) need to integrate with a central legacy gateway that is accessed through the central router.

However, as a rule the linear topology is not very practical. It provides no fault tolerance or alternate path capability, so if a router box or network link goes down, the result is two disparate spheres that cannot communicate with each other. In this respect, if the hardware infrastructure is not overly stable, the whole JMQ application can become quite brittle.

In addition, the routers must be configured to use full routing, since the default is to use normal routing of interest and connection information. Full routing is a smarter form of inter-router communication that has to be enabled for all but the simple topology. It carries a slightly higher overhead since more information needs to be propagated throughout the router network. The default normal routing is optimized for the simple topology, which is common for many applications and tries to minimize the amount of inter-router administrative message traffic.

An example of a messaging system where linear topology fits well is an integrated manufacturing situation. In Figure 1, the left router connects all of the plant floor sensors and machine control applications. The center router provides the connections to the MRP shop floor control modules, possibly as adapters to a legacy shop floor control module in a commercial MRP/ERP system. The right router connects applications that comprise the overall manufacturing scheduling and planning system.

In this scenario, the three application areas reside on different network local loops, and are in various geographic locations (shop floor automation in the plant, shop floor control module in the regional office, and planning at the head office). However, there is a large amount of local message traffic in each of the three areas, making separate routers very applicable. Since the shop floor machine automation on the left does not require a direct connection to the 10,000-foot planning system on the right, the linear topology fits quite nicely.

Interconnected Topologies

Interconnected topologies resemble a network of routers, somewhat similar to a spider's web, which provides at least one alternate path between applications.

Routers keep track of connection information continuously, and if they detect that connectivity is lost to an adjacent router, they route the message on any available alternate paths. Interconnected topologies provide superior fault tolerance and fail-over capability over simple and linear topologies, and are a good choice if the system has stringent availability requirements.

The drawback of interconnected topologies is that the routers maintain and propagate more connection and interest information amongst themselves, which can affect throughput rates. In addition, routers must be configured to use full routing of interest and connection information, since the default is to use normal routing which is optimized for the popular simple topology. Full routing is a smarter form of inter-router communication that must be enabled for all but the simple topology. It carries a slightly higher overhead because more information is propagated throughout the router network.

There are two forms of interconnected topologies: partially and fully. Partially interconnected topologies provide at least one alternate path between all JMQ technology-enabled applications, but every router is not connected to every other router in the system. This means that some messages travel through more than one router to get from the producer application to the destination consumer application, giving them a longer message latency. Figure 2 shows a partially interconnected topology.

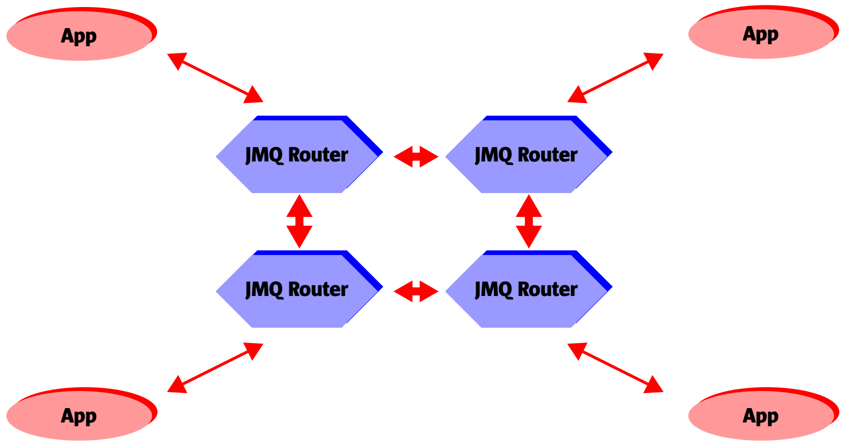

FIGURE 2 Partially Interconnected Topology

In this example, it is clear that the failure of any one link between routers will not disable any of the applications that are running because the routers can find and use an alternate path. Similarly, the failure of any router affects only the applications that are directly connected to it, and these applications can attempt to reestablish connection to an alternate router for recovery purposes.

Fully interconnected topologies connect every router to every other router in the system. When there are many routers, this looks like a child's spirograph drawing. A simple example of a fully interconnected topology is shown in Figure 3.

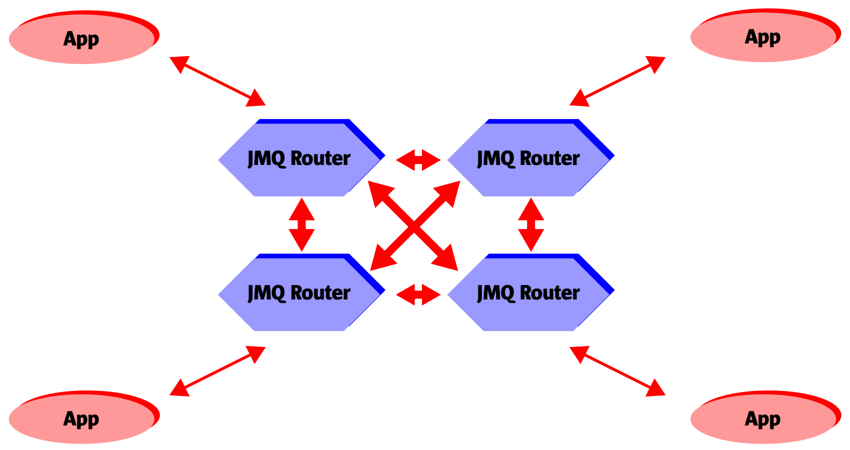

FIGURE 3 Fully Interconnected Topology

Since every router is connected to all the others, application producers and consumers are guaranteed to be one hop apart, reducing message latency and providing consistency between application processes (assuming consistent hardware transport layer latency). Fault tolerance and failover is at its best with a fully interconnected topology, because it provides the maximum amount of alternate message paths between any two applications.

On the surface, it seems that the fully interconnected topology does not need full forwarding/routing turned on (e.g., the –R switch). However, one of the key features of the fully interconnected strategy is its fault tolerance. If a link is lost between two routers, interest/connection information and messages are not propagated properly between the routers at the ends of the broken link unless full forwarding is turned on. Turning off full forwarding (which is the default) for routers is an optimization intended only for the very simple case of one or two routers (e.g., simple topology).

Keep in mind that with the fully interconnected topology, the large number of router interconnections can raise the overhead due to connection and interest propagation between routers. For large, fully interconnected router networks, this overhead is substantial and drastically affects the message and data throughput rates in the system because the number of interconnects and resulting overhead rises as an exponential progression. For this reason, fully interconnected topologies are recommended for situations that have a small number of routers (three to five) and only for application situations that require the maximum fault tolerance/failover and minimal message latency.

In general, partially interconnected topologies provide good fault tolerance/failover capability while minimizing router propagation overhead at the expense of some increased latency between certain pairs of applications, and thus are usually recommended.

For example, in a stock ticker/news distribution system there are many client applications (subscribers to the stock news service) that must connect to routers, easily exceeding the recommended maximum of 100 connections per router. Given the large volume of messages that such a system must process and route to numerous clients, and the need for guaranteed uptime to satisfy subscribers, an interconnected topology is a good fit. It can spread the router workload for stock news/ticker messages to the selectors for each client (e.g., which stocks the client is tracking) over multiple machines, increasing overall throughput of the system while enhancing fault tolerance and limiting router connections. If the stock ticker/news distribution system is a news clipping service, near real-time response is not required. A partially interconnected topology, although introducing extra latency for certain transmissions, is more than adequate. However, if the application is designed to notify customers of stock price fluctuations (e.g. for automated day trading), a fully interconnected topology provides consistent one-hop latency for all messages.

Hierarchical Topologies

Hierarchical topologies typically have routers interconnected in a pyramid formation or tree structure, as illustrated in Figure 4.

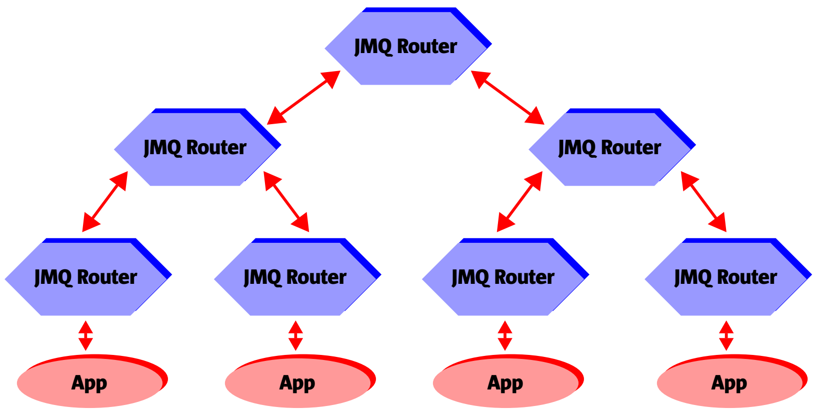

FIGURE 4 Hierarchical Router Topology

Routers higher in the structure act as message concentrators and gateways for the substructures underneath them.

The drawbacks of hierarchical topologies are numerous. In their pure pyramid form, with no horizontal links between routers, they provide no fault tolerance on links, and thus suffer from the same problems as the linear topology. The addition of horizontal links transforms the pure hierarchical topology into either an interconnected or gateway / mixed topology.

In the case of fairly deep router hierarchies, the message latency from an application low in one branch to another application on a different branch can become prohibitive, because many hops may be required to get up the tree to a common router and then back down again.

It is extremely rare for a router process to have only other routers connecting to it and no client applications. It is usually more efficient to interconnect the child routers using some other topology. Since hierarchical topologies are not recommended, an example of the topology is not appropriate.

Gateway and Mixed Topologies

Gateway topologies are very useful when a system needs to account for slow transport links, geographical considerations, or application partitioning requirements.

A gateway topology is used to glue together a number of other topologies, providing a pathway between them. A typical gateway topology, linking two fully interconnected topologies, is illustrated in Figure 5.

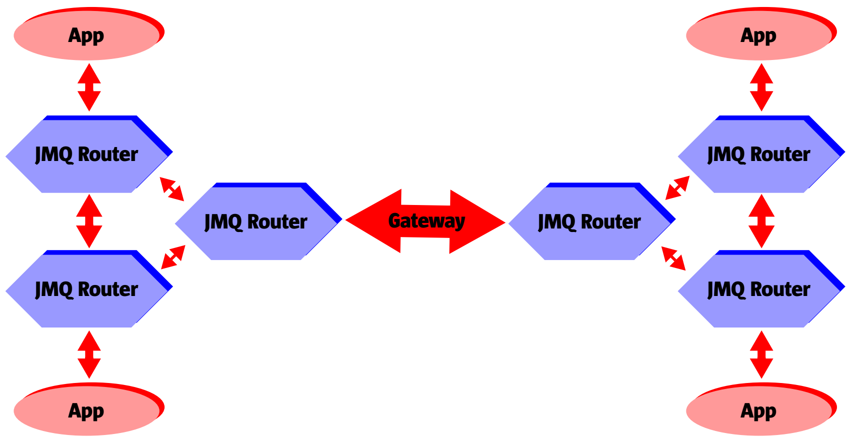

FIGURE 5 Gateway / Mixed Router Topology

If a slow communication link (for example, a leased line with modems) exists between the two other topologies, it is best to use a gateway because the applications need to send only those messages that are common to both sides across the link. Of course, the applications should be structured so that high-volume messages stay on their side of the gateway and are not forwarded across the link. In this way, only essential messages are allowed to consume the limited bandwidth.

Furthermore, using a gateway can eliminate duplicate messages that might get transmitted across a slow link, because routers forward messages to the router that is closest to an interested application. For example, if two consumer applications are connected to two routers on one side of a slow link, a third router on the other side forwards two copies of a message that both applications are interested in: one message to each router on the other side. A gateway forces the message to be transmitted over a single inter-router link, eliminating the duplicate.

If an organization has geographic considerations, a gateway / mixed topology can be very useful. In Figure 5, if the left three routers are located in America and the right three are in Europe, it would not make economic sense to have more than one gateway link between the two systems because much of the message traffic is of local interest to the specific continent. WAN links tend to be slow, expensive, or both, so minimizing traffic across diverse geographical areas is a good idea.

As discussed under the simple topology, performance, throughput, and latency can be improved by partitioning an application into smaller subsystems. Messages that are local to a subsystem are handled with a topology of routers designed to optimize local message traffic, with only the messages that implement the interface between subsystems being transmitted across the gateway. An added benefit of this approach is that the overall application architecture is simplified by performing the partitioning through strictly defined interface messages, much like the precepts of good object-oriented and component engineering practice.

Gateway topologies also make a lot of sense when trying to integrate legacy or non-message-based applications into the overall system. All that is needed is to write an adapter application that interacts with the legacy system to translate legacy events into JMQ technology-enabled messages. This is not to say that adapter applications are necessarily small, simple, or easy to write, but that their existence provides a clean encapsulation of the legacy system for integration purposes.

The drawback of using gateways is that they create a single point of failure for the overall system. In practice this may be unavoidable. Multiple links may be economically unfeasible, as in the case of overseas WAN connections. With partitioned application architectures, the separate subsystems are designed to be fairly autonomous in their operation. For example, losing the link between America and Europe would not impact local company operations much, because most messages are local in nature and are still transported across the local router network.

Furthermore, the use of persistent message queues or topics, along with durable subscriptions for communications across a specific link, can ensure that no messages get lost in the event of a gateway connectivity problem. When the gateway connection is restored, the persistent messages that were queued while the link was unavailable are transmitted to the durable subscribers on the other side of the gateway and processed as required to synchronize the two subsystems.

For more complex application systems, the gateway / mixed topology is the ideal solution because it provides the maximum ability to match router topology to external requirements such as slow links, geography, and simplification of overall application architecture, especially when coupled with other JMS facilities such as persistent message storage and durable subscribers.

An example of the use of gateway / mixed topologies is a distributed conferencing / whiteboard application for a multinational corporation with multiple branches in North America and Europe. Most conferencing messages (e.g., voice / video packets or graphical whiteboard images) are within the continent. So each continent deploys an interconnected topology of routers to supply all of its major regions with appropriate performance and fault-tolerance considerations. Video traffic requires the high performance and low latency provided by a fully interconnected topology. Voice traffic is less demanding and can get by with a partially interconnected topology. Whiteboard images work with almost any connection topology. However, trans-Atlantic conferencing requires a gateway topology to connect the two continents and get the best performance possible out of the slow, expensive WAN connection.

CHAPTER 4

Physical Network and Hardware Platform Considerations

When designing your router topology, take into account the physical characteristics of your network, including: slow links, unreliable links, LANs, and WANs. Most of these considerations have been discussed in some detail in the sections on the various router topologies. In summary, the topology should consider the type of connection (network, modem, interprocess shared memory), the connection bandwidth, and reliability requirements.

In addition, think about the hardware platforms and processors your routers and applications run on. There may be special hardware and devices, as well as unique capabilities of certain machines that will be connected, including video cameras, robots, or shop floor sensors. Other hardware platforms may have special characteristics such as high-speed numeric processors, large databases, or legacy applications.

Along with the "horsepower" or utilization available on each computer platform, consider the user's convenience, keeping in mind that the goal is to optimize overall performance and throughput while minimizing latency.

If your system runs several communicating application processes on a single machine, it is advisable to run a router on the same machine. This enables the router to use interprocess shared memory for messages that pass between applications on the same box, eliminating network latency and bandwidth restrictions.

On fast LANs with high bandwidth, running a router on every machine in a fully interconnected topology, as long as there are not too many routers, is advisable because it provides short, consistent message transmission time with only one hop for any message and provides the best fault tolerance.

In some situations, it may be undesirable or impossible to run both a router and applications on the same machine. For instance, PCs running MS-DOS cannot handle multiple processes, so the application should run on the PC machine and connect to a router on another machine. There are also performance advantages to running a router on a dedicated machine when message volumes are high, throughput is a key requirement, or the application is very resource intensive.

Other considerations include the efficiency and performance of the TCP/IP stack on your hardware platform. Not all machines or operating systems are created equal when it comes to pushing raw TCP/IP packets (which are used by routers) through the stack. Some operating systems and hardware configurations are more efficient at process or thread-context switching, an issue with multithreaded applications and routers — which are multithreaded themselves — or task switching between multiple processes on the same machine.

Router connection and interest propagation overhead tends to produce unwanted performance degradation once connections exceed a certain level per router (100 for PCs, more for larger Sun servers running the Solaris Operating Environment), so it is advisable to stay under this limit. A special flag (-C) can be used to limit the number of connections that a router will accept.

CHAPTER 5

JMQ Router Interconnection Syntax

Interconnecting Routers

To interconnect a router with other routers:

```
irouter -- -s<host1><, host2> …
```

where the new router being started will be connected to routers running on the specified hosts.

For example, to connect a new router to routers running on hosts alpha, beta, and gamma:

```
router -- -salpha,beta,gamma
```

Router Subnet

To start a router on a specific subnet:

```
irouter -- -n<subnet>
```

For example, to start a new router on subnet 3 (the default subnet is 0):

```
router -- -n3
```

Full Routing Mode

To turn on full routing (required for all topologies except simple, 1, and 2 router topology):

```
irouter -- -R
```

Limiting Connections

To limit the number of connections that a router will accept:

```
irouter -- -C<connLicType>
```

For example, to limit a new router to use a 25-connection license:

```
router -- -C50
```

To use an unlimited connection license:

```
router -- -Cunl
```

Specifying -C with no `connLicType` value will list the available connection licenses.

CHAPTER 6

Additional Information

For more information on router command syntax, please refer to the JMQ Deployment Guide.

More information on JMS can be found at:

http://java.sun.com/products/jms/

Detailed information on JMQ can be found at:

http://www.sun.com/workshop/jmq/

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303

1 (800) 786.7638
1.512.434.1511

http://www.sun.com/workshop/jmq/

December 1999