Switch or Broker?

A comparison of two models for Reliable Messaging

by Pieter Hintjens, iMatix Corporation

January, 2006
The Classic View

- The "reliable message broker"
- Big, powerful message broker
- Uses high-availability and transactions
- Queues are durable, persistent, heavy
- Consistent with traditional world view
  - Strong centralization
  - One place to look for all data
  - One server to configure, administer, etc.
Reliable message broker

- App
- Reliable message broker
- Persistent queues
- App
- App
- App
What do we get?

- Point-to-point reliability
  - When we have successfully passed a message to the broker, we can be sure it will be delivered to the recipient.
  - *If there is a recipient...*
- Pretty complex
- Pedantic message delivery
- Distributed transaction processing
- Performance vs. persistence trade-offs
High-availability cluster

App

App

App

App

?
HA reliability challenges

- Guarantee reliability during failover
- Coordination between HA peers
- Transactions between HA peers
- A problem area for most products
- Presumably a solvable problem...
  - But even MQ Series occasionally drops messages
- Broker is now very complex
  - ...and HA is visible in the protocol
What about wide-area networks?
WAN reliability challenges

- Only as reliable as weakest link
- Excludes use of untrusted middlemen
- Excludes use of thin "edge" brokers
- Contrary to modern network design
  - Requires few, expensive boxes
  - Does not scale by adding hardware
  - Reliability through complexity?
The iMatix View

- The "asynchronous message switch"
- Cheap, disposable, stateless devices
- Organized into high-availability pairs
- Queues are transient and memory-based
- Consistent with modern world view
  - Cheap and simple
  - No data to look for
  - Minimal configuration and administration
Asynchronous message switch
What do we get?

- No reliability in protocol or server
  - Messages can be lost en-route
- Very simple
  - Trivial message delivery
  - No transactions
  - Fastest possible performance
  - Message loss is *very* rare
- But...
  - Of course, we need full reliability
Reliability over AMS (RAMS)

- Two main messaging scenarios
  - Data distribution (publish-subscribe)
  - Service requests (request-response)
- Publish-subscribe can be unreliable
  - Data can be resent periodically
- Request-response should be reliable
  - This is the classic MQ scenario
RAMS Schema

Asynchronous message switch

Persistence

App

App

App

App
Rams Technique

- Client API provides two messaging APIs
  - Data distribution
  - Request-response (R-R)
- Reliability is implemented for R-R only
  - Record request in store
  - Send request to service
  - Wait for a matching response
  - If needed, send the request again
  - When response arrives, update store
  - After timeout, alert application to failure
RAMS with a HA cluster
RAMS + HA is simple

- No assumption of reliability in network
- HA is limited to peer-coordination
- No message flow between peers
- Proven solution in production use
- Broker remains relatively simple
- HA is invisible in the protocol
  - Persistence is also invisible
What about wide-area networks?
RAMS + WAN is simple

- No assumption of reliability in network
  - It does not matter how large network grows
- Reliability is orthogonal to network
- Allows use of untrusted middlemen
- Allows use of thin "edge" servers
- Moves towards "messaging in every wall-plug" vision
Why is RAMS nice?

- Protocol remains simple
  - No reliability semantics
- Brokers remain simple & light
  - No persistence, no transactions, no acks
- Reliability is (almost) unilateral
  - No interoperability burden
  - Recipient must detect & discard duplicates
- Reliability is easy to understand
  - Very intuitive, obvious for programmers
RAMS is fast

- Data distribution can be scaled
  - Fanout for high volumes & subscribers
  - Zero overhead for data distribution
- No centralized storage bottleneck
  - Reliable message broker can only run as fast as its data store
  - AMS runs at full speed
  - Persistence cost is spread out across network
Can RAMS do everything?

- No, it cannot
- RAMS is a "90%" solution
  - Specifically for request-response
  - Also solves data distribution optimally
  - Does not do file distribution
  - Does not do other messaging models
  - But... these can be layered on top
- Cannot solve “MQ Series” challenge
How do we implement file distribution?
Consider RAMS as a transport layer
Use only request-response messaging
Construct services around RAMS
File distribution over RAMS:
  Through use of "file broker" application
  RAMS to and from file broker
## Switch vs Broker

<table>
<thead>
<tr>
<th>Switch</th>
<th>Broker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple protocol</td>
<td>Complex protocol</td>
</tr>
<tr>
<td>Simple server</td>
<td>Complex server</td>
</tr>
<tr>
<td>Edge storage</td>
<td>Central storage</td>
</tr>
<tr>
<td>Intelligent API</td>
<td>Simple client API</td>
</tr>
<tr>
<td>WAN friendly</td>
<td>WAN hostile</td>
</tr>
<tr>
<td>Silicon friendly</td>
<td>Silicon hostile</td>
</tr>
<tr>
<td>Unconventional</td>
<td>Conventional</td>
</tr>
</tbody>
</table>
Conclusions

- RAMS has significant advantages
  - Simply, cheap, flexible
  - Compatible with move to hardware
  - Elegant solution for HA & WAN
- Reliable message broker is outdated
  - Solves problems of the past
- RAMS is future-proof design
  - Cheap, scalable, simple
  - Forces standard messaging patterns
Thank you

- For more information please contact the author at ph@imatix.com.