EMI Messaging PT

Lionel Cons
Massimo Paladin

EMI Techinal Forum – Vilnius, 12th April 2011
Messaging Solutions

• Many solutions available
Messaging Solutions Survey

• Paper survey of the options for an EMI Messaging Service available

• EMI wiki
  – Messaging PT > Messaging Solutions Survey
  short-url: http://goo.gl/tcwBT
Executive Summary

- **Protocols**
  - Main: STOMP & AMQP
  - Plus REST for some use cases (lightweight publishing)

- **Solutions**
  - Many exists, 11 considered
  - No best messaging solution yet
  - Different solutions might suit better different use cases
  - ØMQ introduces new brokerless concept
  - Interesting ØMQ + RabbitMQ integration
News of the Messaging World

• STOMP 1.1 has been released
  – Better defined than 1.0
  – Enables new features (virtual hosts, heart-beating, NACK frames...)

• AMQP
  – 1-0 still in validation process
  – 1-0 is incompatible with previous versions
  – iMatix (one of the designer of AMQP) dropped Open AMQ support in favour of ØMQ

• ActiveMQ Apollo is going to be released in the end of 2011

• HornetQ head, Tim Fox, left its team and has been hired by SpringSource to work on RabbitMQ
General Recommendations

• Minimize the amount of code that you write for messaging
  – Re-use existing code
  – Isolate technology independent code from the rest (brokers, destination...)

• Prepare for bad things
  – Messages can get lost
  – Messages can arrive out of order
  – Messages can be delivered multiple times

• Security
  – Do not trust data by default
  – Use cryptography if needed: encryption and signing
Messaging Protocol

• STOMP is the recommended protocol
  – Good enough for most use cases
  – Supported by many messaging solutions
  – Recommended client libraries:
    • Perl: Net::STOMP::Client
    • Python: stomp.py

• Other protocols might bind you to a specific messaging solution

• Although is not a protocol, JMS is probably the best Java client solution
  – JMS is only API
  – Most of the brokers are JMS compliant
  – Changing messaging solution => changing JMS provider
Message

• Header
  – List key/value pairs
  – Contains simple information, like message body metadata
  – Avoid conflict with other keys (i.e. use prefix)
  – Keys: ASCII letters + digits + dots + dashes
  – Values: ASCII printable characters
  – Avoid many keys and long values

• Body
  – Contains data to send
  – JSON recommended for simple applications
    • Well supported by programming languages
    • Widely used
  – Avoid to stick to a custom format
    • Messaging used for software components integration
**Message Size**

- Messages should be small
  - They are copied many times
  - 1KB to 10KB is the optimal range
  - 1MB should be seen as absolute maximum
- Large messages incompatible with high rates
- Too small messages are not efficient
- Applications can easily adjust message size
- Compression can be used to reduce the size
  - But don’t forget about the CPU time
Message Rate

• Real performance is the one measured in a realistic environment
• Establishing a session can be expensive
  – Especially using X.509
  – Try to minimize the number of sessions
  – Long lived connections can be problematic too
• What matters is the total number of messages in and out
  – Topic with 10 subscribers => 11 messages for each incoming message
• 1k msg/s practical maximum on WAN with STOMP
• If messages are persistent the rate should be reduced
• 1KB messages * 1K msg/s is 10Mbit/s at network level
Conclusions

• Messaging is changing fast
• Avoid using specific protocols and features
• Reliability of a Messaging Service does not depend only from the server
  – Clients should be reliable and well written
• Re-use existing code and libraries

• For more information and contacts:
  – EMI wiki > EMI Messaging (http://goo.gl/yykzQ)
Thank you!

EMI is partially funded by the European Commission under Grant Agreement RI-261611