Distributed Software Architectures using Middleware

Robert Nunn

Who am I?

- I am a 4th year undergraduate on the MSci Computer Science program
- Currently working on a distributed publish-subscribe content-based networking system
  - Vaguely related to this talk.
- You can contact me at r.nunn@cs.ucl.ac.uk

Outline

- Distributed systems
- Middleware
- Types of middleware
- Pub-sub systems
- My project: XBN

Distributed Systems

- Components of the system are not all held on the same host
- Hosts are connected by a computer network
- Appears to users as a single, integrated computing facility

Distributed Systems

- Integration of existing systems.
- Increased performance and reliability.
  - Possibly, but depends on what we are aiming for.
- Can be cheaper than a centralised system.
- However: complicated, lots of possible errors, multiple points of failure, network speeds an issue, …

Middleware

- Software sitting between the operating system and the application
- Hides the underlying network protocols from the programmer
- Common environment across platforms and programming languages
- E.g. Microsoft COM, OMG CORBA, Java RMI, etc.

Middleware

- Makes a programmers life easier:
  - Resolves heterogeneity
  - Provides transparency
  - Higher level of abstraction
  - Can deal with (some) failures automagically
- But can add complexity and cost
- May not be appropriate in all circumstances

Types of Middleware

- Transactional middleware
  - Supports transactions
- Message-oriented middleware
  - Supports message exchange
- Procedural middleware
  - Supports remote procedure calls (RPCs)
- Object/Component middleware
  - OO version of procedural middleware

Transactional Middleware

- Supports transactions involving components on different hosts.
- Transactions ensure that operations occur on all hosts or no hosts.
- Assumes servers use two-phase commit protocol (2PC).
  - Keeps entire system in a consistent state.
- Communication can be synchronous or asynchronous

Transactional Middleware

- Advantages:
  - Easy to integrate with database management systems
  - Guarantees consistency
- Disadvantages:
  - Transactions not always needed
  - Marshalling (and, therefore, unmarshalling) needs to be done by programmer

Message-Oriented Middleware

- Lovely for publish-subscribe systems (my project, discussed later) and distributed event notification
- Communication by message exchange
- Asynchronous: Client sends message and carries on, eventually gets a response

Message-Oriented Middleware

- Advantages:
  - Great for group communication
  - Easy to make fault tolerant
  - Client and server decoupled
- Disadvantages:
  - No access transparency
  - Apps have to do marshalling/unmarshalling

Procedural Middleware

- Support remote procedure calls (RPCs)
- Uses an interface definition language (IDL)
- Synchronous actions between one client and one server
- Middleware deals with marshalling and unmarshalling
- Used across multiple platforms and programming languages

Procedural Middleware

- Advantages:
  - Simple for programmers
  - Familiar
  - Bindings for many programming languages
- Disadvantages:
  - No support for multicast or asynchronous communication
  - Not scalable
  - Not fault tolerant

Object/Component Middleware

- OO extension of procedural middleware
- Adds inheritance, references, exceptions, etc.
- Can also support transactions, messaging, synchronous and asynchronous comms, load balancing

Object/Component Middleware

- Advantages:
  - Integrate features of the other forms of middleware
  - Very powerful, flexible, etc.
- Disadvantages:
  - Limited scalability
  - Not always applicable in non-OO environments

Pub-Sub Systems

- Publish-Subscribe can implement a content-based network
- Pub-sub is a clever type of message oriented middleware (MOM)
  - The normal MOM way:
    - Client chooses channels to listen to
    - Messages published to channel
    - Clients receive all messages published to channel

Pub-Sub Systems

- In pub-sub:
  - clients specify what they want to receive
    - I.e. the content they are interested in
  - Publishers send messages to network
  - Network figures out where messages should go

Subscriptions

- A list of restrictions on content of messages
  - E.g. news involving Iraq, changes in share price of BT, …
  - I.e. defines a subset of messages
- A subscriber sends a subscription to the network which then propagates it
- Subscriber only receives messages from the subset it has defined

Publications

- Sent by publishers
- Contain information which is of interest to subscribers
  - Although, it may not be received by anyone!
- Forwarded by dispatchers to all subscribers that should get it

Dispatchers

- Like routers in IP networks
- Routing table contains subscriptions
- If a subscription *matches* a publication then forward it to the subscriber
  - Matches = "publication is in the subset defined by the subscription"
  - Nastiness from set theory involved in creating efficient dispatcher

Pub-Sub Systems

- Advantages:
  - Intuitive method of communication
  - Subscribers are anonymous to publishers and most of the dispatchers
- Disadvantages:
  - Scalability of system vs. expressiveness of subscription language
  - Subscribers are anonymous to publishers and most of the dispatchers

XBN

- XML Based Networking
- Subscriptions are a list of XPath expressions
  - I really use a subset of XPath (don't ask why)
- Publications are XML documents
- P2P arrangement:
  - Publishers look and act like dispatchers
  - Dispatchers look and act like subscribers

XBN

- XPaths are slow to evaluate: need real computers as dispatchers
- Cannot use whole XPath language
  - But still allows for a very expressive subscription language
- Everyone loves XML!

Summary

- Distributed systems are lovely but complicated to build and maintain
- Middleware hides many of the complications of building a distributed system
- Therefore, middleware is also lovely!
- Finally, XBN is also lovely! (well, it will be!)

References

- For information about SIENA, an existing pub-sub system, visit:
  - http://www.cs.colorado.edu/serl/dot/siena.html
- For more information about the types of middleware I have discussed, read:
  - http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/ICSE2000/SOTAR