ECE1770

Trends in Middleware Systems

Introduction & Overivew

Hans-Arno Jacobsen
jacobsen@eecg.toronto.edu
www.eecg.toronto.edu/~jacobsen

Outline

- admin remarks
- motivation & introduction - *what is middleware?*
- approximate lecture-by-lecture course overview

Admin Remarks

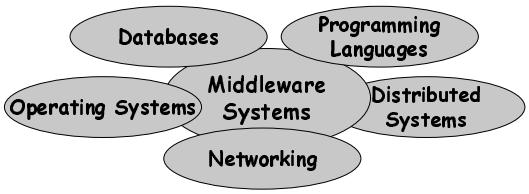
- all this information is available online
  `http://www.eecg.toronto.edu/~jacobsen/ece1770/`
- **all announcements** will be made available via this web site
- most **required reading material** is linked on this site

Middleware

Definition by Example

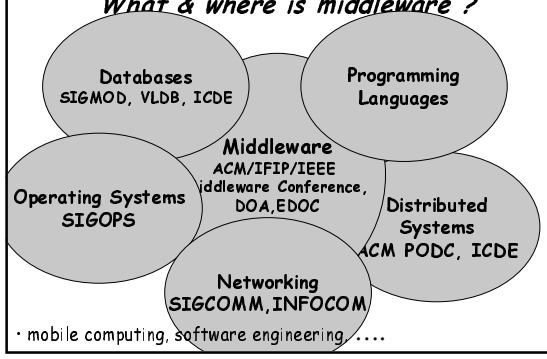
- The following constitute middleware systems or middleware platforms
  - CORBA, DCE, RMI, J2EE (?), Web Services, DCOM, COM+, .Net(?), application servers, …
  - some of these are collections and aggregations of many different services
  - some are marketing terms

What & where is middleware ?



- middleware is dispersed among many disciplines

What & where is middleware ?



- mobile computing, software engineering, ...

Middleware research

- dispersed among different fields
- with different research methodologies
- different standards, points of views, and approaches
- a middleware community is starting to crystalize
- many existing fields/communities are broadening their scope
- "middleware" is still somewhat a trendy or marketing term, but I think it is crystallizing into a separate field - **middleware systems**.
- in the long term we are trying to identify concepts and build a body of knowledge that identifies middleware systesm - much like OS - PL - DS ...

Research Interest of the Middleware Systems Research Group - MSRG

- distributed systems and applications
  - **mobile wireless** application (e.g., **location-based services**)
  - Internet-applications (e.g., **Application Service Providing** and **Web Services**)
  - **peer-to-peer networking & computing** (i.e., architectures like Napster, Gnutella, MojoNation etc.)
  - **sensor networks**
- **design** and **implementation** of **middleware systems**
  - experiment with new design methodologies - *aspects, reflection, open implementation, modularization* - to build middleware platforms
- **our major current thrust areas**:
  - ***Event Notification & Publish/Subscribe***
  - ***Aspect oriented middleware design***

MIDDLEWARE SYSTEMS RESEARCH GROUP

Middleware Systems- Definition

... are **services** to facilitate the **development** and **deployment** of distributed applications in heterogeneous environments.

Examples:

- remote communication mechanisms (CORBA, Java RMI, DCOM - i.e. request broker)
- event notification and messaging services (COSS Notifications, Java Messaging Service)
- transaction services
- naming services (COSS Naming, LDAP)

Middleware

- software technologies to help manage complexity and heterogeneity inherent to the development of distributed systems, distributed applications, and information systems
- layer of software above the operating system and the network substrate, but below the application
- higher level programming abstraction for developing the distributed application
- higher than "lower" level abstractions, such as sockets provided by the operating system
- a socket is a communication end-point from which data can be received or onto which data can be sent

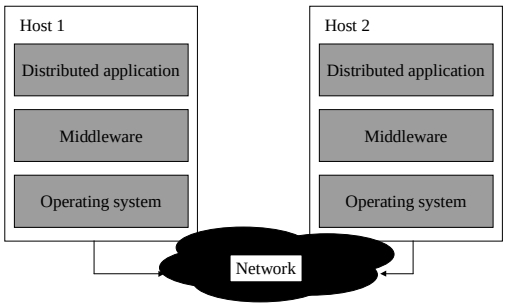
Middleware cont.'d.

- aims at reducing the burden of developing distributed application for developer
- informally called "plumbing", i.e., like pipes that connect entities for communication
- often called "glue code", i.e., it glues independent systems together and makes them work together
- it masks the heterogeneity programmers of distributed applications have to deal with
  - network & hardware
  - operating system & programming language
  - different middleware platforms
  - location, access, failure, concurrency, mobility, ...
- often also referred to as transparencies, i.e., network transparency, location transparency etc.

Middleware cont.'d.

- an operating system is "the software that makes the hardware usable"
- similarly, a middleware system makes the distributed system programmable and manageable
- bare computer without OS could be programmed, so could the distributed application be developed without middleware
- programs could be written in assembly, but higher-level languages are far more productive for this purpose
- however, sometimes the *assembly-variant* is chosen - WHY ?

Middleware



Categories of middleware

- we don't follow the exact taxonomy suggested by Bakken;
- categories
  - remote invocation mechanisms
    - e.g., DCOM, CORBA, DCE, Sun RPC, Java RMI, Web Services ...
  - naming and directory services
    - e.g., JNDI, LDAP, COSS Naming, DNS, COSS trader, ...
  - message oriented middleware
    - e.g., JMS, MQSI, MQSeries, ...
  - publish/subscribe systems
    - e.g., JMS, various proprietary systems, COSS Notification

Categories cont.'d.

- (distributed) tuple spaces
  - (databases) - I do not consider a DBMS a middleware system
  - LNDA, initially an abstraction for developing parallel programs
  - inspired InfoSpaces, later JavaSpaces, later JINI
    - entities can read and write from a tuple space
    - they are de-coupled in space, i.e, do not have references to each other (i.e, are a priori anonymous)
    - they are de-coupled in time, i.e, do not have to be up simultaneously
    - compare this to remote invocation !
    - closely related to publish/subscribe
- transaction processing system (TP-monitors)
  - implement transactional applications, e.g.e, ATM example

These categories mirror our lectures and will be treated in greater detail in the specific lecture module.

Middleware platforms & standards

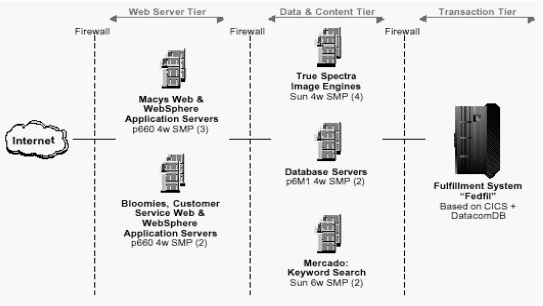
- in the "old" days simply an RPC mechanisms defined by the Open Software Foundation (now Open Group)
- evolved into DCE (Distributed Computing Environment) RPC + Security + Time + Events +... also by the OSF (1980s, a standard with reference implementations)
- CORBA (Common Object Request Broker Architecture) by the Object Management Group (89 - present, a standard with many implementations)
- DCOM (Distributed Component Object Model) Microsoft's de facto standard for building distributed applications on Windows-based platforms (199? - ??, de facto standard with implementations on Windows (mostly))
- >NET by Microsoft (2000-present, lot's of tools etc.)
- Web Serices "communication over the Web", a set of standards by a number of key players

In the end it is all about getting data from point A to point B !

Motivation

- the following case studies haven been taken from a presentation at the DeveloperWorks Conference 2002
- these are real world applications
- for us the importance is the overall architecture and their characteristics **not** the exact detail of each case study
- we want to understand what an information system and a distributed applications really is

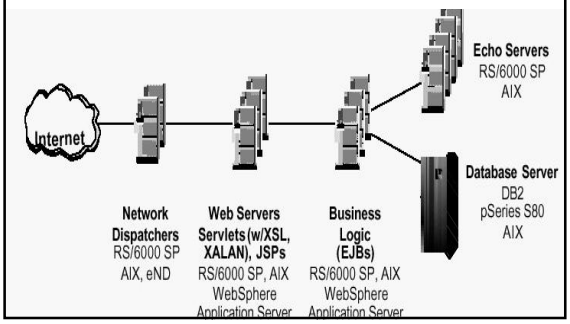
Online system of a department store



Observations & Remarks

- architecture runs web-site of a department store
  - user may browse for items
  - user may search for items
  - user may order items
- small web-server farm
- multi-tired architecture
- individual tiers are shielded by firewalls
- individual tiers are comprised of multiple servers
- each server fulfills concrete task
  - e.g., serves images, keyword search, security, buy & sell transaction, …
  - implemented by dedicated stand-alone software

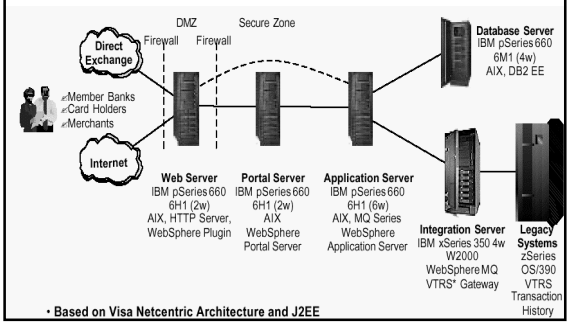
Online trading architecture



Observations & Remarks

- architecture runs web-site of an online broker
  - view stock ticks, track stocks, buy & sell stock
- high-degree of redundancy in architecture
  - achieved by replicating servers
  - achieves high availability & high throughput
- a top-level domain can be mapped to multiple target domains depending on load, geographic region, customer profile (e.g., gold customers better service (faster response))
- www.ebroker.cad could be mapped to
  - www.machine1.ebroker.cad
  - www.machine2.ebroker.cad …

A Credit card company's IS



Observations & Remarks

- integration of legacy system and legacy data
  - e.g., systems and infrastructre that pre-dates the Web and the Internet
  - data that has been collected for years

Numbers from a consumer payment service (credit card)

- about 2000 network end points in 100 countries
- 30+ billion transactions processed annually worldwide
- transaction latency should be less than 1 sec.
- 7 x 24 availability

Sample real world requirements

- design for scalability
  - increasing traffic & react to peak loads
- plan for growth
  - more content, more hardware
- design pages for performance
  - dynamic content
  - content fetched from DBs, other apps
- web site personalization on a per user basis
- maximize site availability

Industries that deploy very large information systems

- health care & hospitals
- banking & finance
- stock exchange
- insurance
- governments
- ...

Banks & financial institutions

- run same application on redundant platforms
  - different OS, different DBs, different communication mechanisms, on different hardware
  - this is to decrease the risk of failure of one of these components
  - *a maintenance nightmare*

Issues (non-exhaustive list)

- development of a new information system, one design should be re-deployable
- a deployed system can not be replaced, it must be extended
- integration of legacy systems
- hardware, software, development languages & paradigms change
- data in system must be accessible, shared across architecture
- ...

More conceptually speaking

- large and complex systems that consists of **physically distributed** computing nodes
- nodes are often not homogeneous
  - ***different** OS, **different** hardware platform,...*
- we say IS runs in a **heterogeneous** environment
- *information system infrastructure* is the software that allows us to build such applications
- we also call this the ***middleware system*** (in analogy to operating system, database system ...)

Questions we are asking & trying to answer about midddleware systems

- What techniques are available to build such systems ?
- What software abstractions exist ?
- What (programming) models exist ?
- What really happens ?
  - *What is the current best practice ?*
  - *What are current standards ?*
  - ***What are current research trends ?***
- What are the invariant concepts underlying ever changing standards and new products ?

In this course we will answer some of these questions and aim at getting a better feel for what the best practice is by studying concepts of existing standard solutions, by looking at standards as case studies and examples, and by focusing on current RESEARCH in the area

Course objective

- **identify** and understand middleware concepts
- learn about current best practice
- learn about the state-of-the-art research on middleware systems

Course structure

- presentation of background material, concepts, and algorithms
  - based on broader (if available), tutorial-like research papers and reports
- focus on detailed algorithm or concept
  - based on research paper
- overview of a "special topic" area by students in a focused presentation at end of lecture

This is the ideal structure - we may deviate here and there, if the adequate reading is not available.

Course Overview

(approximately 12 weeks)

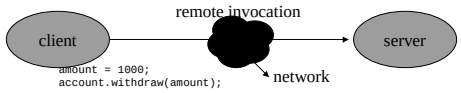
ECE1770 - approximate course outline

1. Introduction and overview
2. Reference models, modeling and the "making" of middleware
3. Remote communication models, their implementation, and novel design techniques
4. Naming in distributed systems
5. Messaging and message oriented middleware

Outline cont.'d.

6. Event notification and publish/subscribe
7. Data distribution services and tuple spaces
8. Transaction processing and transaction processing monitors
9. Mediators and database access
10. (Web-)Application Servers

Communication and invocation at the application level I + II



- what is a remote invocation ?
- how can the integer `amount` be transferred over the network, be re-assembled on the server side ?
- does the client wait for a response from the server ?
- what if the network goes down while the invocation is in progress ?
- how is this programmed ? … in Java ? in XYZ ?
- what tools exist to do remote invocations ?
  - **examples: CORBA, Java RMI, and Web Services** (current trend)
- what are the limitations of remote invocation, are there any at all ?

Expert Topic

- Overview of Web Services architecture
- Middleware for Grid computing

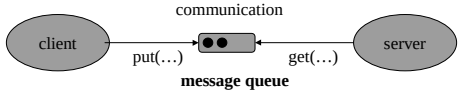
Naming in distributed systems

- how does the client **find** the server in a distributed system ?
- how is a name `www.ebroker.cad` **resolved** in the network, i.e, how is the server that serves pages for ebroker found ?
- can a service be **discovered** by looking for its properties ?
- difference between domain name and address ?
- how many addresses may be assigned ?
- what abstractions and services are available to do this ?
  - **our examples: DNS, JNDI, LDAP**

Expert Topic

- Naming in peer-to-peer systems

Messaging and message oriented middleware



- what is different in this model from remote invocations ?
- what different characteristics may be associated with the message queue ?
- how is this model used for communication, how can a the operation account.withdraw(1000) be modeled ?
- what are merits and de-merits of this model ?
- **examples: JMS, MQSeries**

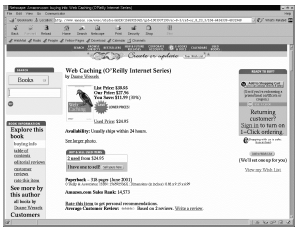
Expert Topic

- Messaging and transaction processing

Event notification and publish/subscribe

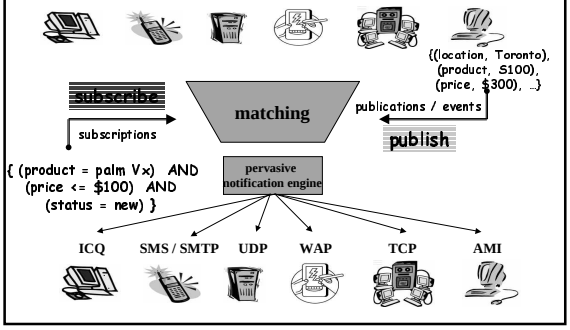
Amazon to Chapters to you ...

Monday, October 10th in Cyberia

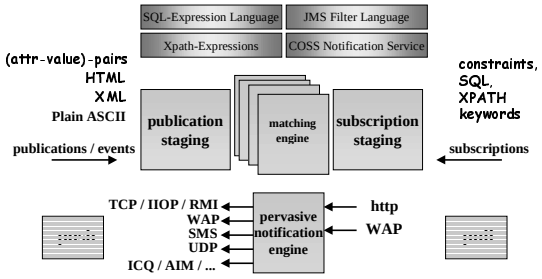


Thursday, November 15th, in Toronto

Event Notification & Publish/Subscribe



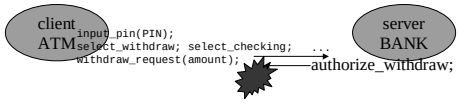
ToPSS - Toronto Publish/Subscribe



Expert Topic

- Real time issues in publish/subscribe

Transaction processing and transaction processing monitors



- A server or network crash in the middle of a sequence of operations !
- what should happen ?
- how can a sequence of operations be grouped ?
- what models exist ?
- how is this done in networked environments, with multiple servers involved in one such sequence ?

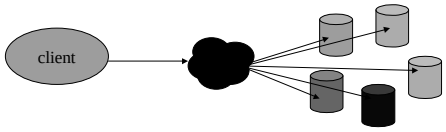
Expert Topic

- Messaging and transactions

Data Distribution and Tuple Spaces

- the management of (persistent) data in a distributed space
  - control systems
  - active spaces
- the coordination of (distributed) activities
- a model for parallel computing
- more specialized abstractions
- no standard solutions (yet)
- a variety of prototypes in existence

Mediators and data access



- how are databases (over the network) accessed ?
- how are multiple, heterogeneous databases accessed ?
- what problems are involved ?
- examples: ODBC & JDBC and mediators

Expert Topic

- Mediators for XML data

(Web-)Application Servers

- what is involved in an online transaction, e.g., buying a book at Amazon or buying and selling stocks on eTrade ? I.e what happens on the server site ?
- where and how is the business logic implemented that is behind these transactions ?
- how are the application that are behind this business logic managed ?
- how has this technology evolved over the past 20 years ?

Today's reading.

Toda's Reading

- **Middleware** David E. Bakken. To appear in: Encyclopedia of Distributed Computing, Kluwer Academic Publisher. (cached locally).
- **Managing Complexity: Middleware Explained** Andrew T. Campbell, Geoff Coulson, and Michael E. Kounavis IT Professional, Vol. 1, No. 5, September/October 1999. (cached locally).