

Middleware in Context

Prof. Dave Bakken

Cpt. S 464/564 Lecture

Auxiliary Material (not from text)

August 24 and beyond, 2011

Material: <http://www.eecs.wsu.edu/~bakken/middleware.pdf>

(564 only): <http://gridstat.net/publications/TR-GS-013.pdf>

Context: (Most) Technology Marches On

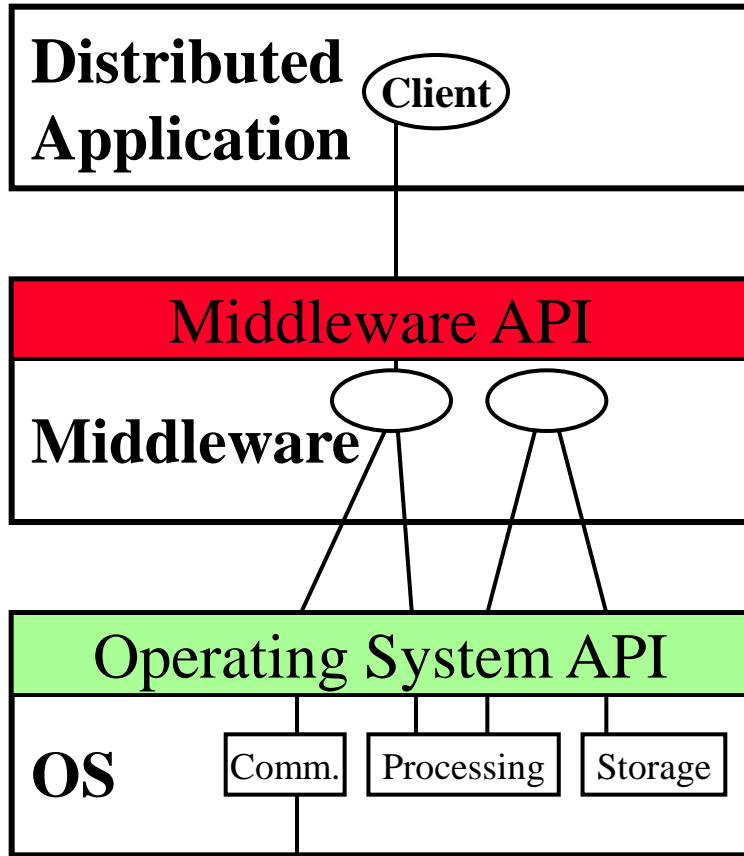
- Hardware technology's progress phenomenal in last few decades
 - Moore's Law
 - Metcalf's Law
 - Graphics processing power
- Software technology's progress is much more spotty
 - “Software crisis”
 - Yet SW is a large and increasing part of complex apps/systems!
- Apps and systems are rapidly becoming (more) networked
 - Oops, distributed software is much harder yet to get right...
- Middleware a promising technology for programability of distributed systems
 - Also fertile grounds for adaptivity and dependability....

Why Middleware?

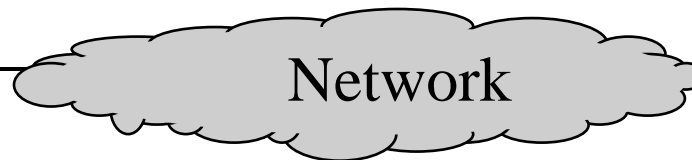
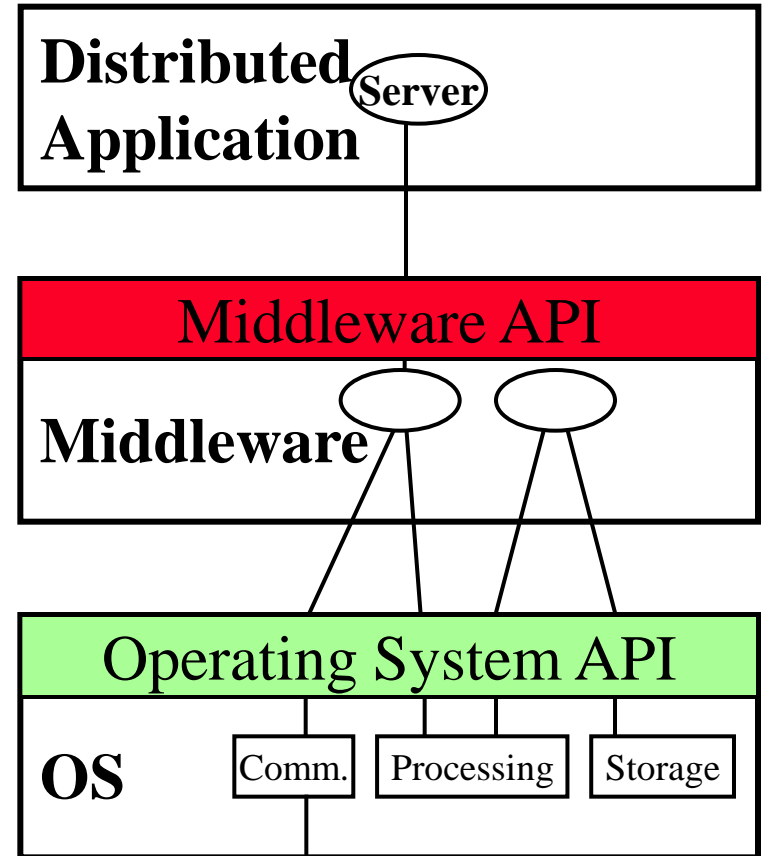
- Middleware == **“A layer of software above the operating system but below the application program that provides a common programming abstraction across a distributed system”**
- Middleware exists to help manage the complexity and heterogeneity inherent in distributed systems
- Middleware provides higher-level building blocks (“abstractions”) for programmers than the OS provides
 - Can make code much more portable
 - Can make them much more productive
 - Can make the resulting code have fewer errors
 - Analogy — MW:sockets \approx HOL:assembler
- Middleware sometimes is informally called “plumbing”
 - Connects parts of a distributed application with “data pipes” and passes data between them

Middleware in Context

Host 1



Host 2



Network

Middleware Benefit: Masking Heterogeneity

- Middleware's programming building blocks mask heterogeneity
 - Makes programmer's life much easier!!
- Kinds of heterogeneity masked by middleware (MW) frameworks
 - All MW masks heterogeneity in network technology
 - All MW masks heterogeneity in host CPU
 - Almost all MW masks heterogeneity in operating system (or family thereof)
 - Notable exception: Microsoft middleware (*de facto*; not *de jure* or *de fiat*)
 - Almost all MW masks heterogeneity in programming language
 - Noteable exception: Java RMI
 - Some MW masks heterogeneity in vendor implementations
 - CORBA best here

Middleware Benefit: Transparency

- Middleware can provide useful transparencies:
 - Access Transparency
 - Location transparency
 - Concurrency transparency
 - Replication transparency
 - Failure transparency
 - Mobility transparency
- Masking heterogeneity and providing transparency makes programming distributed systems much easier to do!

Programming with Middleware

- Programming with Middleware
 - Do not have to learn a new programming language! (Usually)
 - Use an existing one already familiar with: C++, Java, C#, Ada, (yuk) Visual Basic, (yuk) COBOL
- Ways to Program with Middleware
 1. Middleware system provides library of functions (Linda, others)
 2. Support directly in language from beginning (Java and JVM)
 3. External Interface Definition Language (IDL) that “maps” to the language and generates local “proxy”

Kinds of Middleware

- Distributed Tuples: (a, b, c, d, e)
 - Relational databases, SQL, relational algebra
 - Linda and tuple spaces
 - JavaSpaces (used by Java Jini)
- Remote procedure call (RPC)
 - make a function call look local even if non-local
- Message-Oriented Middleware (MOM)
 - messages and message queues
- Data/topic-based publish-subscribe
- Distributed Object Middleware
 - Make an object method look local even if non-local
 - CORBA
 - DCOM/SOAP/.NET
 - Java RMI

Kinds of Middleware (cont.)

Different middleware systems encapsulate and integrate the different kinds of resources with varying degrees:

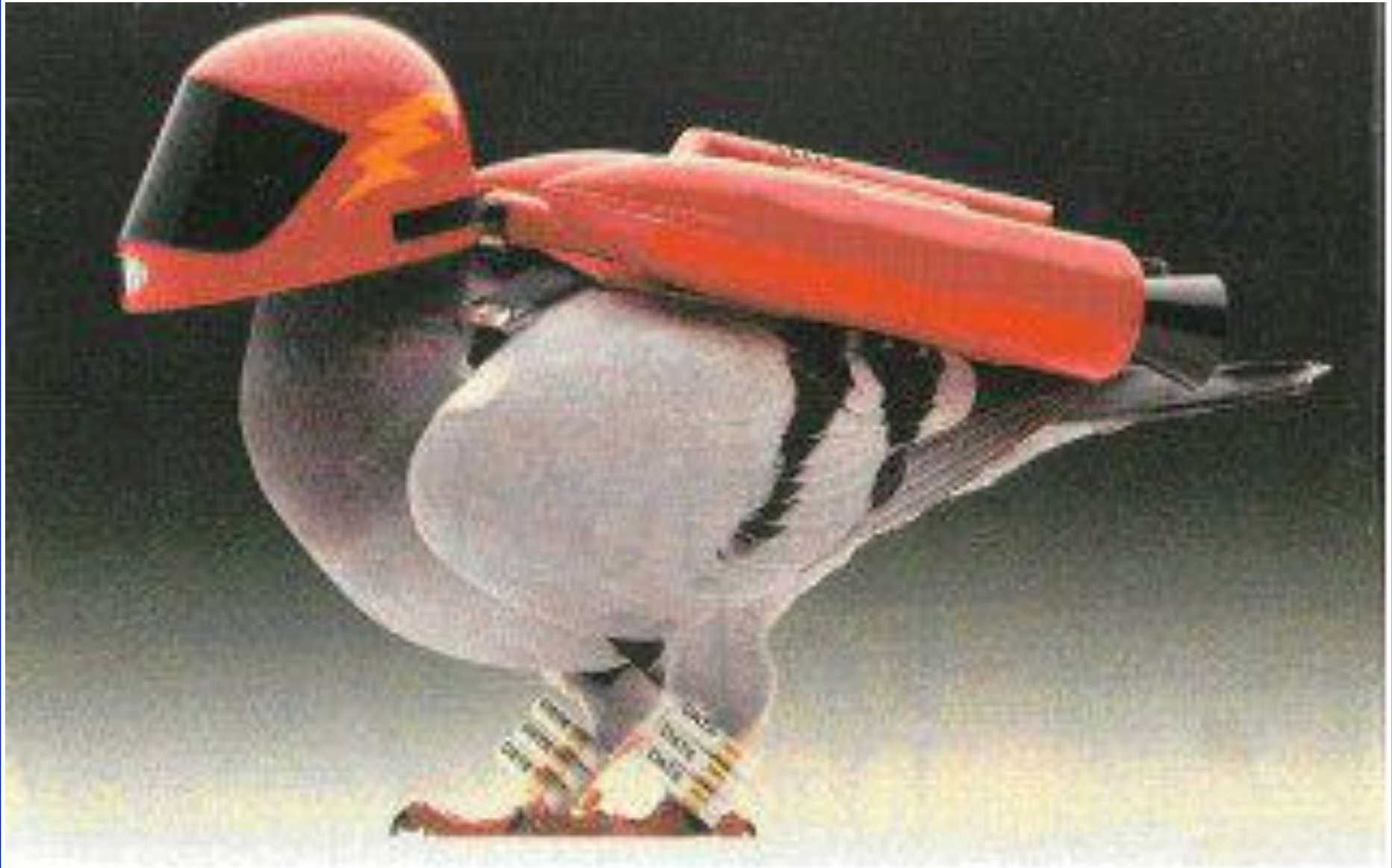
Middleware Category	Communication	Processing	Storage
Distributed Tuples	Yes	Limited	Yes
Remote Procedure Call	Yes	Yes	No
Message-Oriented MW	Yes	No	Limited
Data/Topic Based	Yes	No	Limited
Distributed Objects	Yes	Yes	Yes

For many (non-database) applications, and supporting adaptation, distributed object middleware is better because it is more general
•But pub sub and tuples are more decoupled which can help

Middleware and Legacy Systems

- Legacy systems are a huge problem (and asset) in industry and military domains!
- Middleware often called a “glue” technology: integrated “legacy” components
 - Much distributed programming involves integrating components, not building them from scratch!
- Middleware’s abstractions are general enough to allow legacy systems to be “wrapped”
 - Distributed objects are best here because more general
 - End result: a very high-level “lowest common denominator” of interoperability

Multi-Layered Middleware



One Middleware Layering Taxonomy (BBN/Schantz)

APPLICATIONS

DOMAIN-SPECIFIC SERVICES

COMMON MIDDLEWARE SERVICES

DISTRIBUTION MIDDLEWARE

INFRASTRUCTURE MIDDLEWARE

OPERATING SYSTEMS & PROTOCOLS

HARDWARE

- **Domain-Specific Services**
 - Services and APIs tailored to (and reusable only within) certain domains (health care, telecommunications, etc)
 - Examples: CORBA Domain Interfaces, Boeing Bold Stroke architecture
- **Common MW Services**
 - Adds high-level, domain-independent reusable services for events, fault tolerance, security,
 - Examples: CORBAServices, Eternal
- **Distribution MW**
 - Provides rich distributed object model that supports much heterogeneity and transparency
 - Examples: CORBA, .NET., Java RMI
- **Infrastructure MW**
 - Encapsulates core OS Comm. and concurrency services (sometimes enhances them too)
 - Examples: JVM (and other VMs), ACE, group comm.

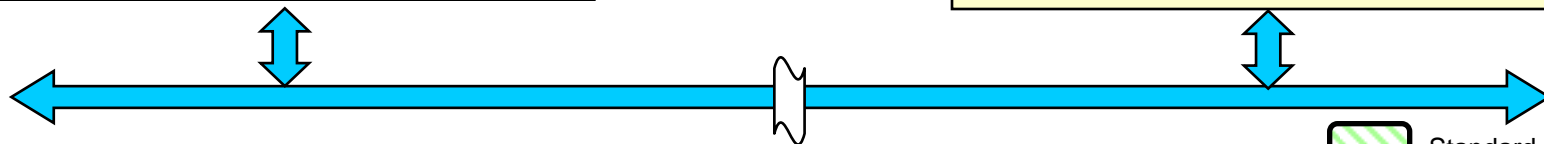
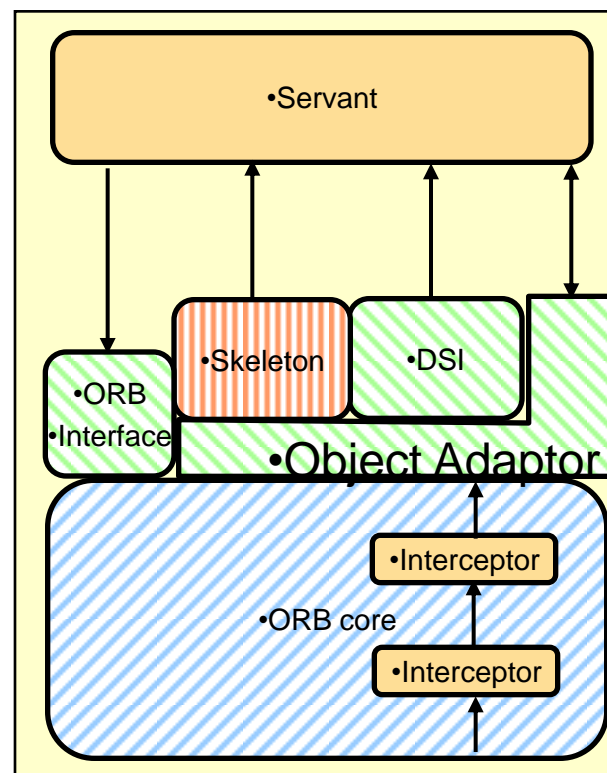
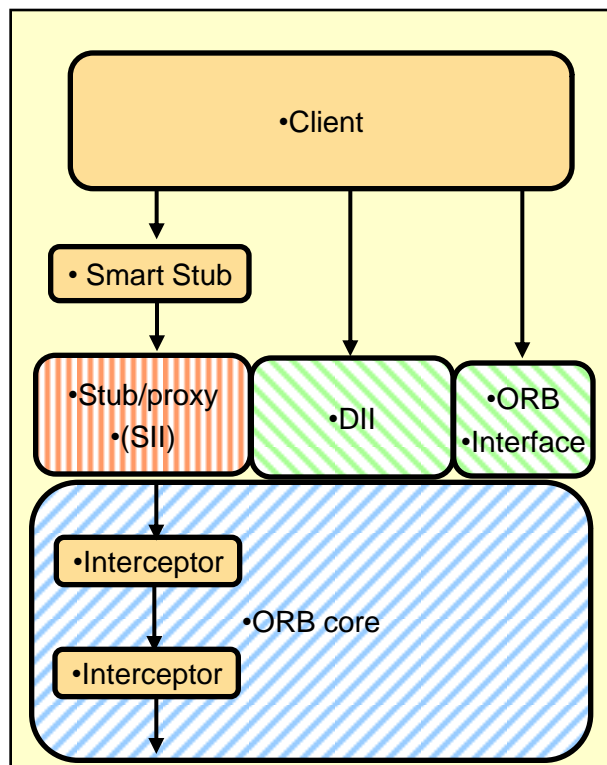
(Figure courtesy of D. Schmidt)




CORBA and System Builders' Hooks

Interface
Repository

IDL Compiler

Implementation
Repository



-  Standard Interfaces
-  IDL-generated
-  ORB-Specific